Data Mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données

Nicolas Pasquier

N° d'ordre : D.U. 1192
EDSPIC : 209

UNIVERSITÉ CLERMONT-FERRAND II

ÉCOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

THÈSE

présentée par

Nicolas PASQUIER

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : INFORMATIQUE

Data Mining : Algorithmes d'Extraction et de Réduction des Règles d'Association dans les Bases de Données

Soutenue publiquement le 31 janvier 2000 devant le jury :

M. Alain QUILLIOT	**Président**
M. Georges GARDARIN	**Rapporteur**
M. Nicolas SPYRATOS	**Rapporteur**
Mme Rosine CICCHETTI	**Examinateur**
M. Jean-Marc PETIT	**Examinateur**
M. Michel SCHNEIDER	**Examinateur**
M. Gerd STUMME	**Examinateur**
M. Lotfi LAKHAL	**Directeur de thèse**

Remerciements

Cette thèse a été réalisée au sein du Laboratoire d'Informatique et de Modélisation des Organisations et des Systèmes de l'université Blaise Pascal - Clermont-Ferrand II. Ce travail a été financé par une allocation de recherche du Ministère de l'Education Nationale, de la Recherche et de la Technologie du gouvernement Français. Je tiens à remercier toutes les personnes qui ont contribué de manière directe ou indirecte à l'aboutissement de ce travail : En premier lieu, je remercie vivement Monsieur Lotfi Lakhal qui m'a permis d'effectuer cette thèse sous sa direction. J'ai particulièrement apprécié son enthousiasme, sa disponibilité et sa gentillesse. Je remercie Messieurs Georges Gardarin et Nicolas Spyratos d'avoir accepté d'être les rapporteurs de ma thèse. Je les remercie pour l'attention avec laquelle ils ont lu et évalué ce mémoire ainsi que pour les remarques et critiques constructives qu'ils m'ont adressées. Je remercie également Monsieur Alain Quilliot, Madame Rosine Cicchetti et Messieurs Jean-Marc Petit, Michel Schneider et Gerd Stumme qui ont accepté d'être les autres membres du jury. Je remercie Messieurs Michel Schneider et Alain Quilliot qui m'ont permis d'effectuer cette thèse dans de bonnes conditions. Je remercie Messieurs Yves Bastide et Rafik Taouil avec lesquels j'ai pris grand plaisir à travailler. Je remercie tous les membres du Laboratoire d'Informatique et toutes les personnes que j'ai cotôyé au sein de l'université Blaise Pascal : Alain, Amina, les Christophe, Eric, Farouk, Hélène, Housni, Kitsana, Jeanne-Marie, Jérôme, Madame Seguy, Marie-Laure, Marlène, Michel, Michelle, Nadine, Nicolaï, Nicolas, Patrice, Patricia, Philippe, Quentin, Raoul, Stéphane, Valérie et Yahia. Des pensées particulières vont à tous mes amis : Alex, Bob, Celine, Driss, Fany, Géraldine, Jean-Christophe, Jérôme, Nathalie et les trois Patrick qui m'ont soutenu durant ces années. Enfin, mille merci à mon père, ma mère, mes grand-parents et mon frère pour le soutien et l'aide qu'ils m'ont apportés durant mes études et sans qui ce travail n'aurait pas pu avoir lieu. Que tous ceux et celles que j'ai oubliés n'en recoivent pas moins ma gratitude.

Nicolas Pasquier, LIMOS

Résumé

L'extraction de connaissances dans les bases de données, également appelé data mining, désigne le processus non trivial permettant d'extraire des informations et des connaissances utiles qui sont enfouies dans les bases de données, les entrepôts de données (data warehouses) ou autres sources de données. Dans ce mémoire, nous traitons des problèmes de la génération efficace des règles d'association et de la pertinence et de l'utilité des règles d'association extraites. Une règle d'association est une implication conditionnelle entre ensembles d'attributs binaires appelés items. Dans l'ensemble des travaux existants, l'extraction de règles d'association est décomposée en deux sous-problèmes qui sont la recherche des ensembles fréquents d'items et la génération des règles d'association à partir de ces ensembles. Le premier sous-problème, dont la complexité est exponentielle dans la taille de la relation et qui nécessite de parcourir à plusieurs reprises celle-ci, constitue la phase la plus coûteuse en termes de temps d'exécution et d'espace mémoire. Nous proposons une nouvelle sémantique pour le problème de l'extraction des règles d'association basée sur la connexion de Galois d'une relation binaire finie. Utilisant cette sémantique, nous démontrons que les ensembles fermés fréquents d'items constituent un ensemble générateur non redondant pour les ensembles fréquents d'items et les règles d'association. Nous proposons deux nouveaux algorithmes, nommés Close et A-Close, permettant l'extraction des ensembles fermés fréquents d'items, à partir desquels les ensembles fréquents d'items et les règles d'association peuvent être dérivés sans accéder au jeu de données. Les résultats expérimentaux démontrent que ces algorithmes permettent de réduire les temps d'extraction et l'espace mémoire nécessaire dans le cas de jeux de données constitués de données denses ou corrélées. Utilisant la sémantique définie, nous proposons d'améliorer la pertinence et l'utilité des règles d'association extraites en limitant l'extraction à des bases pour les règles d'association. Nous adaptons pour cela les bases pour les règles d'implication définies en analyse de données et nous définissons de nouvelles bases constituées des règles non redondantes d'antécédents minimaux et de conséquences maximales à partir des ensembles fermés fréquents. Nous proposons également des algorithmes efficaces de génération de ces bases.

Mots-Clefs : Extraction de connaissances dans les bases de données, data mining, connexion de Galois, opérateurs de fermeture, treillis des ensembles fermés fréquents, règles d'association, bases pour les règles d'association.

Nicolas Pasquier, LIMOS

Table des matières

Liste des figures

Liste des tables

Liste des algorithmes

Notations

Description des symboles utilisés dans la thèse.

Relation d'ordre :

$Join(E)$	plus petit majorant des éléments de E
$Meet(E)$	plus grand minorant des éléments de E
$\leq$	relation d'ordre
$\lessdot$	relation de couverture

Théorie des ensembles :

$\cup$	union
$\bigcup$	union généralisée
$\cap$	intersection
$\bigcap$	intersection généralisée
$\subseteq$	inclusion
$\subset$	inclusion sans égalité
$\in$	appartenance
$\setminus$	soustraction
$\|E\|$	cardinalité de l'ensemble E
2^E	ensemble des parties de E

Logique des prédicats :

$\vee$	disjonction
$\wedge$	conjonction
$\neq$	négation
$\Longrightarrow$	implication
$\Longleftrightarrow$	équivalence

Fonctions et applications :

$f \circ g(E) = g(f(E))$	composition de fonctions

Chapitre 1

Introduction

Sommaire

1.1 Extraction de connaissances dans les bases de données

L'extraction de connaissances dans les bases de données (KDD[1]) désigne le processus non-trivial d'extraction d'informations implicites, précédemment inconnues et potentiellement utiles concernant les données stockées dans les bases de données [PSF91]. Les travaux en ce domaine sont motivés par l'évolution très rapide des techniques de génération (telles que les techniques de lecture des codes barres des articles achetés en supermarchés) et de stockage des données (augmentation de la capacité et diminution des coûts des disques durs par exemple) qui ont permis la création par de nombreux organismes de volumineuses bases de données concernant leurs activités. Ce sont les organismes commerciaux (banques, supermarchés, organismes de vente par correspondance, etc.), les administrations et les organismes scientifiques, de communication et industriels qui utilisent désormais d'importantes bases de données dans le cadre de leurs activités. L'idée sous-jacente au KDD est qu'il est possible d'extraire des informations cachées dans ces masses de données, utiles pour l'aide à la décision, la gestion des informations, l'optimisation des requêtes, le contrôle de processus, etc. Ces informations sont des règles, des concepts, des régularités, des anomalies et des modèles qui sont utiles, intéressants et compréhensibles du point de vue de l'utilisateur [FPSSU96]. L'analyse par exemple de bases de données de transactions de ventes de supermarchés permettra d'étudier les comportements des clients, en fonction de quoi il sera possible de réorganiser les rayons afin d'améliorer les ventes. En analysant des données d'organismes de vente par correspondance, il est possible de regrouper les clients selon certains critères ce qui permettra ensuite de limiter les coûts des « mailing » en définissant de manière plus précise les clients potentiels. Les information provenant de l'analyse de données d'organismes médicaux offriront un support supplémentaire pour les recherches médicales. D'une manière plus générale, ces informations permettront d'apporter une aide pour l'amélioration

[1]Le terme originel anglais « Knowledge Discovery in Databases » a été introduit par Piatetsky-Shapiro en 1989. Afin de simplifier la lecture, nous utilisons dans la suite de la thèse l'abréviation « KDD » pour désigner l'extraction de connaissances dans les bases de données.

de l'efficacité de l'organisme dans ses activités. Le KDD constitue donc un sujet de recherche majeur pour la communauté des bases de données [SSU91, SAD+93].

Le KDD est un processus semi-automatique et itératif, constitué de plusieurs étapes allant de la sélection et la préparation des données jusqu'a l'interprétation des résultats, en passant par la phase de recherche des connaissances : le *data mining*[2] [FPSS96a]. Les différentes étapes de ce processus sont présentées dans la figure 1.1. La phase de data mining désigne l'application aux données préparées, de méthodes

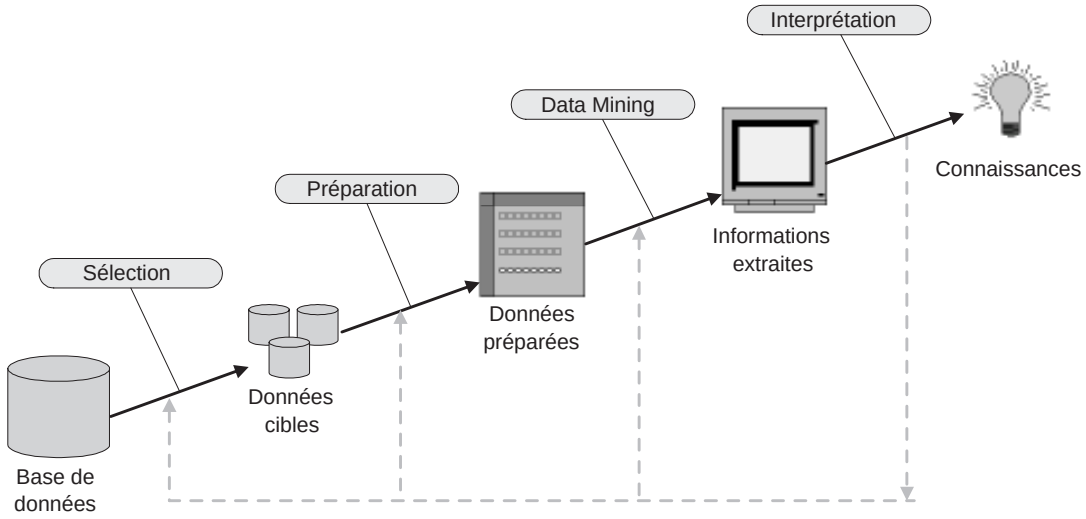


FIG. 1.1 – Étapes du processus de KDD.

et techniques qui fournissent un ensemble d'informations sur les données. Le type d'informations fournies dépend de la finalité du processus, c'est à dire du problème traité. On peut distinguer les cinq principaux problèmes suivants :

Règles d'association Le problème de l'extraction de règles d'association fut introduit par Agrawal et al. dans [AIS93b]. Ce problème, développé à l'origine pour l'analyse de bases de données de transactions de ventes, a pour but de découvrir des relations significatives entre les données de la base. Étant donné une base de données de transactions, chacune constituée d'une liste d'articles achetés par un

[2]Le terme « data mining » est fréquemment utilisé comme synonyme de « KDD ». Nous l'utilisons dans ce mémoire dans son sens originel : il désigne la phase d'extraction des informations, à partir des données préparées, dans le processus de KDD.

client, une règle d'association est une relation d'implication $X \rightarrow Y$ entre deux ensembles d'articles X et Y. Cette règle indique que les transactions qui contiennent les articles de l'ensemble X ont tendance à contenir les articles de l'ensemble Y. Un exemple de règle d'association est : « 40% des transactions qui contiennent les articles *pain* et *vin* contiennent également l'article *fromage* ; 5% des transactions contiennent ces trois articles ». Dans cette règle, 40% est la *confiance* de la règle et 5% est le *support* de la règle. Ces mesures de précision, empruntées aux statistiques, permettent de déterminer les règles d'association qui sont significatives : la confiance indique la proportion de clients qui ont acheté l'article *fromage* parmi ceux qui ont achetés les articles *pain* et *vin*, et le support indique la proportion de clients qui ont achetés les trois articles. Le problème consiste à extraire l'ensemble des règles d'association dont le support et la confiance sont au moins égaux à des seuils minimaux de support et de confiance définis par l'utilisateur. Les règles d'association ont été utilisées avec succès dans de nombreux domaines, parmi lesquels l'aide à la planification commerciale, l'aide au diagnostic et en recherche médicale, l'amélioration des processus de télécommunications, de l'organisation et de l'accès aux sites Internet, l'analyse d'images, de données spatiales, géographiques et statistiques. Le problème de l'extraction de règles d'association et ses application est décrit dans la section 1.2.

Classification Le problème de la classification des données consiste en la détermination des propriétés communes aux objets d'une base de données et leur placement, selon un modèle de classification, dans un ensemble de classes prédéfinies. Le modèle de classification est construit en choisissant un échantillon significatif d'objets de la base de données, que l'on appelle *jeu d'essai*, dans lequel chaque objet (appelé *exemple*) va être associé à un nom de classe. L'objectif est tout d'abord d'analyser les caractéristiques des exemples du jeu d'essai afin de développer un modèle descriptif de chaque classe. Ces modèles (règles de classification, arbres de décision, etc.) sont ensuite utilisés afin de classifier les objets de la base de données. Ce problème concerne de nombreuses applications comme le diagnostic médical, la gestion de stocks, la gestion des données semi-structurées et le ciblage de clientèle. Une société de vente par correspondance par exemple, pourra réduire les coûts d'un « mailing » en spécifiant une classe de clients correspondant à l'objet de celui-ci et en ne s'adressant qu'aux clients qui auront été classifiés dans cette

classe. La classification des données a été étudié dans les domaines des statistiques [EP96, MST94], des réseaux de neurones [Lip87, LSL95], de l'apprentissage numérique [BFOS84, Cat91, Qui93], des systèmes experts [Wid95], et est un important problème du KDD [AGI+92, FPSSU96, HCC92].

Clustering L'objectif du clustering, également appelé *classification non supervisée*, est de fractionner l'ensemble hétérogène d'objets de la base de données en un certain nombre de sous-ensembles plus homogènes, appelés *clusters*. Cela signifie que les clusters doivent contenir des objets qui partagent un haut degré de similarité (maximisation de la similarité intra-cluster) et que ces clusters doivent être suffisamment larges (minimisation de la similarité inter-cluster). La similarité des objets est en général mesurée en termes de distance géométrique entre les objets. La création de clusters répond à des besoins très proches de ceux de la classification : elle permet de fractionner les objets de la base en groupes d'objets ayant un comportement similaire, lorsque les classes ne sont pas prédéfinies. Les méthodes de clustering peuvent également être utilisées lors de la phase initiale de la classification des données : les clusters fournissent une base utile pour la définition des classes. Le problème du clustering a été étudié dans les domaines des statistiques [CS96], de l'analyse de données [CR93], de l'apprentissage numérique [Fis87, MS83], des bases de données spatiales [BKSS90, EKX95] et du KDD [AGGR98, EKSX96, NH94, SG99, Wai98, Wai99, ZRL96].

Séries chronologiques Le problème de la découverte de séries chronologiques fut introduit par Agrawal et Srikant dans [AS95], puis étendu dans [SA96b]. Dans de nombreuses bases de données, une étiquette de temps est associée a chaque donnée de la base (objet ou attribut). Dans le cas des bases de données de transactions de ventes par correspondance, une date, un numéro de client et une liste d'articles achetés sont associés à chaque transaction. Le problème peut alors se formuler ainsi : étant données les listes ordonnées des transactions de chaque client, comment déterminer les séquences d'articles qui sont récurrentes dans les listes. La recherche d'épisodes fréquents dans une séquence, qui est une variante de ce problème, a été étudiée par Mannila et al. dans [MTV95] puis étendue dans [MT96a]. La découverte des séries chronologiques trouve des applications dans le suivi de clients, le mailing, la recherche de séquences génétiques [WMS+94], l'aide au diagnostic, la catégorisation

de documents [FD95, LAS97], l'étude des secousses telluriques, etc.

Généralisation La généralisation des données a pour objectif de fournir un ensemble réduit de tuples qui décrivent les objets de la base de données à un niveau d'abstraction plus élevé. Deux approches peuvent être distinguées : l'approche par « data cube » [GHQ95, HRU96] (bases de données multi-dimensionnelles, On-Line Analytical Processing, etc.) et l'approche par induction orientée attributs [HCC93, HF96, Mic83]. L'idée de l'approche par data cubes est de matérialiser les résultats de calculs coûteux qui sont fréquemment requis, plus particulièrement les calculs qui utilisent les fonctions d'agrégation (compte, somme, moyenne, etc.). Dans l'approche par induction orientée attributs, la taxonomie des attributs de la base de données est utilisée afin de généraliser les objets et fusionner les doublons (tout en maintenant leur décompte). Le résultat peut ensuite être utilisé afin, par exemple, de générer des règles d'association ou des séries chronologiques généralisées [MT96b].

Les solutions apportées à certains de ces problèmes dans les domaines des statistiques, de l'apprentissage numérique, de l'analyse de données et des systèmes experts sont applicables seulement pour des volumes de données réduits (de l'ordre du millier d'objets et de la centaine d'attributs) du fait de la complexité exponentielle en temps et en espace de ces problèmes. Dans le KDD, ce sont de grandes bases de données qui sont traitées : de plusieurs dizaines de milliers à plusieurs millions d'objets, et plusieurs milliers d'attributs en général [MCPS93]. Ceci pose d'important problèmes d'efficacité que l'échantillonnage ne permet pas de résoudre entièrement [Toi96a]. Il est donc nécessaire de développer de nouvelles méthodes, sous des contraintes fortes d'efficacité en temps d'exécution, pour l'extraction d'informations prenant en compte l'importance des volumes de données [CHY96, FPSS96b, Man97]. Ce constat constitue le fondement des recherches en data mining.

1.2 Extraction de règles d'association

Introduit par Agrawal et al. dans [AIS93b], l'extraction de règles d'association est l'un des principaux problèmes du KDD. Ce problème fut développé pour l'analyse de bases de données de transactions de ventes, chacune constituée d'une liste d'articles

achetés, afin d'identifier les groupes d'articles achetés le plus fréquemment ensemble. Une règle d'association sera par exemple : « Les clients qui achètent de la bière ont tendance à acheter des cacahuètes ». Une telle règle d'association permettra de définir une stratégie commerciale dans le but d'augmenter les ventes, en plaçant par exemple la bière et les cacahuètes dans le même rayonnage du magasin. Afin de ne générer que les relations significatives entre les ensembles d'articles, des mesures d'utilité (le support) et de précision (la confiance), empruntées aux statistiques, sont associées à chaque règle d'association. Les règles générées sont celles dont le support et la confiance sont supérieurs ou égaux à des seuils minimaux définis par l'utilisateur en fonction de ses objectifs et du type de données traitées.

1.2.1 Applications

L'extraction de règles d'association a pour but d'identifier les relations significatives entre les données des bases de données. Les relations ainsi identifiées peuvent être utiles pour de nombreux organismes commerciaux, scientifiques, industriels et de gestion de l'information, afin d'améliorer leurs résultats dans leurs activités. Plusieurs systèmes de KDD utilisant l'extraction de règles d'association ont été utilisés pour des applications réelles dans diverses domaines. Nous présentons dans la suite une liste non exhaustive des applications dont les résultats ont put être améliorées par l'analyse des règles d'association extraites.

Planification commerciale [AIS93b, Ass90, FPSSU96, PSF91] L'identification des articles achetés fréquemment ensemble apporte une aide importante dans le placement des articles. Un problème proche de celui-ci est la définition de catalogues. Les règles d'association permettent aux sociétés de vente par correspondance de déterminer quels articles il est préférable de placer sur la même page d'un catalogue. Ces informations sont aussi utilisées afin de déterminer quels articles en promotion pourront inciter les clients à effectuer d'autres achats. Dans le cas de transactions de ventes dans lesquelles le client est identifié, les règles d'association permettent de définir des catalogues personnalisés en se basant sur les achats précédents du client. Elles permettent également de réduire les coûts des mailing en identifiant les clients les plus susceptibles de répondre à chaque mailing selon leurs achats précédents.

Réseaux de télécommunications Les bases de données d'alarmes détectées dans les réseaux de télécommunications sont constituées de rapports de situations anormales dans les composants des réseaux. Classiquement, ce sont plusieurs milliers d'alarmes qui sont détectées chaque jour, plusieurs milliers de types d'alarmes pouvant être distingués. Dans ce cadre, les règles d'association ont été utilisées avec succès dans le système TASA [HKM+96, KMT97] pour le filtrage des alarmes non informatives, l'identification des causes d'anomalies, et la détection et la prédiction d'anomalies. Les règles d'association ont également été utilisées pour la prédiction d'incidents dans les processus de télé-maintenance, afin de limiter les coûts des interventions manuelles et d'améliorer la qualité du service [AMS97].

Recherche médicale [MYGS91, OO98, PMS97, PSM94] La plupart des organismes médicaux (hôpitaux, laboratoires d'analyse, cabinets médicaux, etc.) stockent systématiquement les informations relatives à leurs patients dans des bases de données. Ces informations sont les résultats de consultations auprès des médecins, les résultats de mesures indiquant la condition du patient et des données sur l'évolution de la condition du patient pendant le traitement. L'extraction de règles d'association dans ces bases de données permet d'apporter une aide au diagnostic en identifiant les symptômes ou maladies précurseurs d'une maladie, une aide dans la définition de traitements en déterminant les symptômes ultérieurs ou les effets secondaires possibles, l'identification de populations à risque vis à vis de certaines maladies, etc. Les règles d'association ont également été utilisées dans le cadre de la prédiction de résultats d'analyses médicales [AMS97]. Les règles d'association extraites ont permis d'identifier les analyses fréquemment pratiquées sur les mêmes patients, et de prédire les résultats de certaines analyses par combinaison de caractéristiques des patients et de résultats d'autres analyses.

Analyse de données spatiales Les bases de données spatiales sont largement utilisées dans les systèmes d'information géographiques, en cartographie, en astronomie et pour les études de l'environnement. Elles stockent des informations spatiales et non-spatiales relatives aux objets (forme, dimensions, positions, couleurs, température, etc.) qui occupent un espace. Du fait du développement des outils automatiques d'acquisition et des outils d'acquisition à distance leur nombre et leur volume ne cesse de croître. Afin de découvrir des informations utiles enfouies dans

ces bases de données, des techniques d'analyse de grands volumes de données, telle que l'extraction de règles d'association, sont nécessaires. Les règles d'association extraites depuis ces données définissent des relations entre des caractéristiques spatiales ou non-spatiales des objets. Elles ont été utilisées, notamment dans le système GeoMiner [HKS97], pour l'aide à la prédiction d'événements naturels (éruptions, tremblements de terre, ouragans, etc.) et à l'aménagement du territoire, pour la prévision météorologique et les études biologiques, démographiques et géographiques [EKS97, KH95, KHA98, MM93].

Multi-média et Internet Des quantités croissantes de données de diverses types (images, audio, vidéo, etc.), appelées données multi-médias, sont stockées dans des bases de données dont le nombre ne cessent d'augmenter. L'extraction de règles d'association à partir de données multi-médias a donné lieu à de nombreuses études, principalement dans le cadre de l'analyse d'images [Czy96, OO98, ZHL+98]. Les applications concernent la reconnaissance militaire, le filtrage des données parasites, la prévision météorologique, l'imagerie médicale, l'aide dans les enquêtes criminelles, etc. De même, un grand nombre de ressources sont accessibles par le réseaux Internet et un nombre important d'accès à ces données sont réalisés chaque jour par des millions d'utilisateurs. La taille et le nombre croissants des sites Internet entraînent d'importants besoins d'outils pour la réorganisation de ces sites en fonction des cheminements des usagers, l'aide à la navigation dans les systèmes de gestion d'informations, la recherche et la sélection des sites (moteurs de recherche), etc. L'extraction de règles d'association à partir des historiques des accès par les usagers aux ressources des sites Internet ont été utilisées dans ce cadre pour l'aide à la conception et l'organisation des sites [CMS97, CMS99, EDV97].

Analyse de données statistiques L'analyse de données statistiques constitue un défi important pour le KDD de par sa difficulté et l'intérêt des informations qui peuvent être extraites de ces données. La difficulté provient de la nature des données statistiques, qui sont fortement corrélées et denses [SW85], ce qui pose d'important problèmes d'efficacité [BMS97]. L'intérêt tient au nombre d'applications pouvant bénéficier de l'analyse des données statistiques qu'elles utilisent. Les organismes financiers, de recherche et les administrations stockent de nombreuses données de ce type (résultats de recensements, de sondages et d'études par exemple). L'analyse

de ces données constitue une part importante de l'activité de ces organismes, et les règles d'association peuvent constituer des indicateurs utiles dans ce cadre.

L'extraction de règles d'association a été utilisée pour de nombreuses autres applications, car elle constitue un module important des systèmes de KDD commercialisés, parmi lesquels on peut citer les systèmes Intelligent Miner [Mac98] de IBM (USA) et DBMiner [HFW+96, HCC+97] de l'université Simon Fraser (Canada).

1.2.2 Définition du problème

Dans la suite, nous présentons la formalisation du problème proposée par Agrawal et al. dans [AIS93a] puis étendue dans [AS94]. Soit $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ un ensemble de m littéraux, appelés *items*[3]. Soit $\mathcal{B} = \{t_1, t_2, \ldots, t_n\}$ une base de données de n transactions, chaque transaction t_i étant constituée d'un sous-ensemble $I \subseteq \mathcal{I}$ d'items et identifiée par un identifiant unique appelé TID. Un sous-ensemble $I \subseteq \mathcal{I}$ de taille k est appelé un k-itemset. Une transaction t_i « contient » un itemset I si et seulement si $I \subseteq t_i$. Le support d'un itemset I est le pourcentage de transactions de $\mathcal{B}$ qui contiennent I :

$$support(I) = \frac{|\{t \in \mathcal{B} \mid I \subseteq t\}|}{|\{t \in \mathcal{B}\}|}.$$

Un itemset dont le support est supérieur ou égal au seuil minimal de support défini par l'utilisateur est appelé *itemset fréquent*[4]. Une règle d'association est une implication de la forme $I_1 \rightarrow I_2$ entre deux itemsets $I_1, I_2 \subseteq \mathcal{I}$ telle que $I_1 \cap I_2 = \varnothing$. La confiance d'une règle d'association $r : I_1 \rightarrow I_2$ est la probabilité conditionnelle qu'une transaction contienne I_2 sachant qu'elle contient I_1 :

$$confiance(r) = \frac{support(I_1 \cup I_2)}{support(I_1)}.$$

[3]Le terme « item », traduction en anglais de « article », a pour origine les bases de données de transactions de ventes. Afin de simplifier la lecture, nous appelons par la suite « item » un littéral et « itemset » un ensemble de littéraux. La phase de définition des items utilisés pour l'extraction de règles d'association est décrite dans la section 1.2.3.1.

[4]Dans la littérature anglaise, les itemsets fréquents sont appelés « frequent itemsets » ou « large itemsets » indifféremment.

Le support d'une règle d'association $r : I_1 \rightarrow I_2$ est égal au support de l'union des itemsets qui la constituent :

$$support(r) = support(I_1 \cup I_2).$$

Étant donné une base de données de transactions $\mathcal{B}$, le problème de l'extraction des règles d'association dans $\mathcal{B}$ consiste à déterminer l'ensemble des règles d'association dont le support et la confiance sont au moins égaux à des seuils minimaux de support *minsupport* et de confiance *minconfiance* définis par l'utilisateur. Ce problème peut être décomposé en deux sous-problèmes :

1. Déterminer l'ensemble des itemsets fréquents dans $\mathcal{B}$, c'est à dire les itemsets dont le support est supérieur ou égal à *minsupport*.
2. Pour chaque itemset fréquent I_1, générer toutes les règles d'association de la forme $r : I_2 \rightarrow I_1 - I_2$ telles que $I_2 \subset I_1$ et dont la confiance est supérieure ou égale à *minconfiance*.

Le premier sous-problème a une complexité exponentielle car étant donné un ensemble d'items de taille m, le nombre d'itemsets fréquents potentiels est 2^m. Il est donc nécessaire de développer des méthodes efficaces d'exploration de cet espace de recherche exponentiel [GKMT97, MT97, ZO98]. Le deuxième sous-problème est exponentiel dans la taille des itemsets fréquents, car pour un itemset fréquent I, le nombre de règles d'association non triviales qui peuvent être générées est $2^{|I|} - 2$. Toutefois, la génération des règles d'association à partir des itemsets fréquents ne nécessite aucun balayage de la base de données et les temps de calcul de cette génération sont faibles devant les temps nécessaires à la découverte des itemsets fréquents. Le problème de la réduction des temps de réponse de l'extraction des règles d'association se réduit donc au problème de l'optimisation de la découverte des itemsets fréquents.

1.2.3 Étapes de l'extraction de règles d'association

L'extraction de règles d'association est un processus itératif et interactif [FPSSU96]. Ce processus peut se décomposer en quatre phases qui sont représentées dans la figure 1.2. Les connaissances de l'utilisateur concernant le domaine d'application sont nécessaires lors des phases de pré-traitement, afin d'assister la sélection et la préparation des données, et de post-traitement, pour l'interprétation et l'évaluation des

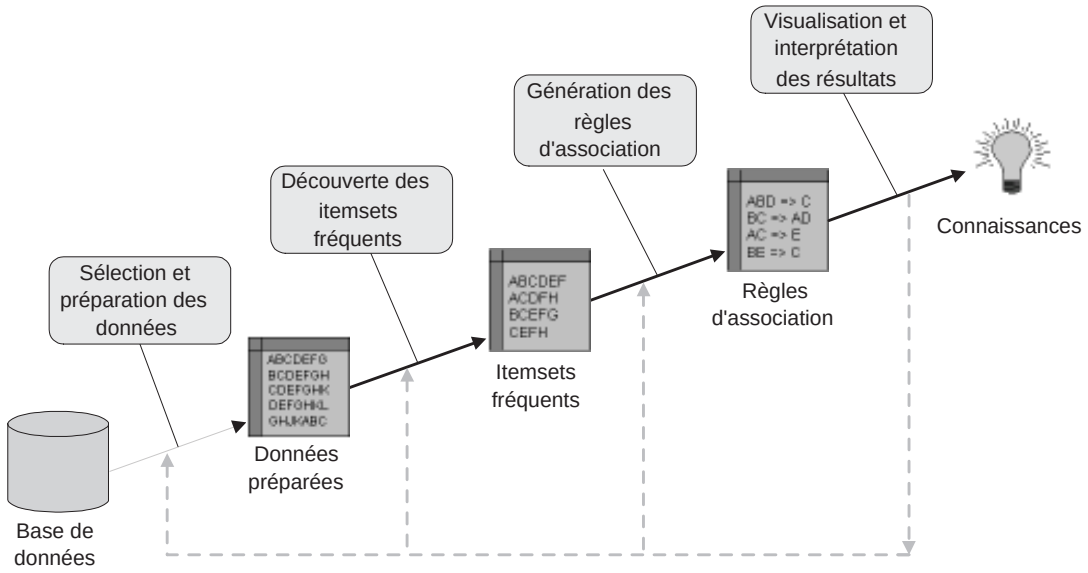


FIG. 1.2 – Étapes du processus d'extraction de règles d'association.

règles extraites. En fonction de l'évaluation des règles extraites, les paramètres utilisés lors des précédentes phases (critères de sélection et préparation des données et seuils minimaux de support et de confiance) peuvent être modifiés avant d'effectuer à nouveau l'extraction des règles d'association, ceci afin d'améliorer la qualité du résultat.

1.2.3.1 Sélection et préparation des données

L'extraction de règles d'association peut être effectuée à partir de bases de données de divers types, telles que des bases de données relationnelles, transactionnelles, spatiales, multi-média, temporelles, orientées objets, etc. Cette phase consiste en deux étapes principales, la sélection des données de la base qui permettront d'extraire les informations intéressant l'utilisateur et la transformation de ces données en un contexte d'extraction. Cette transformation est nécessaire afin qu'il soit possible d'appliquer les algorithmes d'extraction de règles d'association sur diverses types de données [CHY96] et de permettre d'accroître l'efficacité de l'extraction des itemsets fréquents qui doit être réalisée dans des temps raisonnables.

Définition 1.1 (Contexte d'extraction de règles d'association)
Un contexte d'extraction de règles d'association est un triplet $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ *dans*

lequel $\mathcal{O}$ et $\mathcal{I}$ sont des ensembles finis d'objets et d'items respectivement et $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{I}$ est une relation binaire entre les objets et les items. Un couple $(o, i) \in \mathcal{R}$ dénote le fait que l'objet $o \in \mathcal{O}$ est en relation avec l'item $i \in \mathcal{I}$.

Afin d'extraire les règles d'association pertinentes du point de vue des attentes de l'utilisateur, il n'est souvent pas nécessaire de considérer l'ensemble des données de la base dans le processus d'extraction. La première étape de cette phase consiste donc à sélectionner l'ensemble des attributs de la base de données qui permettront d'obtenir des règles utiles du point de vue de l'utilisateur dans le cadre de l'extraction réalisée. La sélection d'un sous-ensemble de l'ensemble des attributs de la base de données permet d'autre part d'améliorer l'efficacité de l'extraction en limitant la taille du contexte d'extraction et donc le coût des balayages de ce dernier.

Lorsque l'ensemble des attributs utiles est défini, les valeurs des attributs, c'est a dire les couples (attribut,valeur), sont transformés en items (littéraux). Chaque item du contexte d'extraction correspond à une valeur ou un ensemble de valeurs (intervalle de valeurs numériques, ensemble de valeurs catégoriques, etc.) d'un des attributs de la base de données sélectionnés. La transformation des données sélectionnées qui sont des données discrètes ou continues en données binaires présente deux avantages. Cela permet d'une part, d'accroître l'efficacité du processus d'extraction [SBMU98] et d'autre part, d'améliorer la pertinence des règles d'association extraites relativement aux attentes de l'utilisateur. Considérons par exemple l'ensemble suivant de règles d'association extraites à partir d'un jeu de données de recensement : {"18 ans $\rightarrow$ sans emploi" (support 2%, confiance 15%), "19 ans $\rightarrow$ sans emploi" (support 1,75%, confiance 15,1%), ..., "24 ans $\rightarrow$ sans emploi" (support 1,80%, confiance 14,9%)}. La transformation des valeurs de l'intervalle [18, 24] de l'attribut "age en années" en un item "moins de 25 ans" permettra d'obtenir une règle unique "moins de 25 ans $\rightarrow$ sans emploi" (support 10%, confiance 15%) dont le support est plus élevé et qui est plus utile que l'ensemble précédent de règles pour l'analyse du taux de chômage selon les tranches d'ages de la population par exemple.

Exemple 1.1 Un contexte d'extraction de règles d'association $\mathcal{D}$ constitué de six objets, chacun identifié par son *OID*, et cinq items est représenté dans la table 1.1. Ce contexte[5] est utilisée comme support pour les exemples dans la suite du mémoire.

[5]Dans la suite, nous utilisons indistinctement les appellations « contexte d'extraction de règles

OID	Items			
1	A	C	D	
2	B	C	E	
3	A	B	C	E
4	B	E		
5	A	B	C	E
6	B	C	E	

TAB. 1.1 – Contexte d'extraction de règles d'association $\mathcal{D}$.

Étant donnée un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, un objet $o \in \mathcal{O}$ contient un itemset $l \subseteq \mathcal{I}$ si tous les items $i \in l$ sont en relation avec l'objet o dans la relation binaire, c'est à dire si $\forall i \in l$ nous avons $(o, i) \in \mathcal{R}$.

Selon la nature et l'origine des données de la base de données, de nombreux autres problèmes doivent également être pris en considération. Ces problèmes concernent entre autres les données incomplètes (valeurs manquantes, etc.), les données redondantes (résultats de calculs stockés pour des raisons de performances, etc.), les données erronées ou incertaines et la taille du jeu de données qui peut imposer la sélection d'un échantillon, afin d'obtenir des temps de réponse acceptables, qui doit être représentatif des données [MCPS93]. La nature de certains types de données, telles que les données spatiales ou les données multi-média par exemple, peut également imposer des pré-traitements particuliers. La résolution de ces problèmes est importante pour la suite du processus d'extraction et pour l'utilité du résultat et la phase de sélection et préparation des données a fait l'objet de nombreuses études [CMS99, Joh97, OO98, SA96a].

1.2.3.2 Découverte des itemsets fréquents

Étant donnée un contexte d'extraction de règles d'association $\mathcal{B}$, la découverte des itemsets fréquents est un problème non trivial car le nombre d'itemsets fréquents potentiels est exponentiel en fonction du nombre d'items du contexte $\mathcal{B}$.

d'association », « contexte » et « relation binaire ».

Définition 1.2 (Ensemble des itemsets fréquents)
Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$. *Étant donné un seuil minimal de support* minsupport, *l'ensemble* F *des itemsets fréquents dans* $\mathcal{B}$ *est :*

$$F = \{l \subseteq \mathcal{I} \mid l \neq \varnothing \ \wedge \ \mathit{support}(l) \geq \text{minsupport}\}.$$

Dans le cas d'un ensemble d'items $\mathcal{I}$ de taille m, le nombre d'itemsets potentiellement fréquents est 2^m. Ces itemsets forment le *treillis des parties* de $\mathcal{I}$, également appelé *treillis des itemsets* de $\mathcal{I}$, dont la hauteur est $m + 1$. La définition du treillis des itemsets est présentée dans la section 2.1. L'itemset fréquent $\varnothing$ n'est en fait pas considéré lors de la recherche car les règles d'association générées à partir de ce dernier ne sont pas des règles informatives.

Exemple 1.2 Le treillis des parties de l'ensemble d'items $\mathcal{I}$ du contexte $\mathcal{D}$ est représenté dans la figure 1.3. L'ensemble $\mathcal{I}$ contenant cinq items, ce treillis contient 32 itemsets et sa hauteur est égale à six.

La phase découverte des itemsets fréquents constitue la phase la plus coûteuse en temps d'exécution de l'extraction de règles d'association du fait de cet espace de recherche de taille exponentielle dans le nombre d'items et de la nécessité de réaliser des balayages du contexte. Ces balayages, nécessaires afin d'évaluer les supports des itemsets qui permettront de déterminer lesquels sont fréquents et de définir les confiances des règles d'association, constituent des opérations très coûteuses en temps d'exécution. Une approche simpliste consiste à tester le support de chacun des itemsets du treillis. Cette approche, bien que ne nécessitant qu'un seul balayage du contexte, est impraticable lorsque le nombre d'items est grand. Dans le cas d'applications réelles d'extraction de règles d'association, le nombre d'items considérés est en général de l'ordre du millier et il est impossible d'appliquer cette méthode. De plus, le nombre d'itemsets fréquents dans le treillis (qui dépend du seuil minimal de support choisi) est en général faible devant le nombre total d'itemsets et la plupart des opérations effectuées lors de l'application de cette méthode sont inutiles car elles concernent des itemsets infréquents.

Plusieurs méthodes permettant de réduire l'espace de recherche de cette phase ainsi que le nombre de balayages du contexte réalisés ont été proposés. Parmi ces méthodes, celles ayant permis d'améliorer de manière significative cette phase sont présentés dans le chapitre 2.

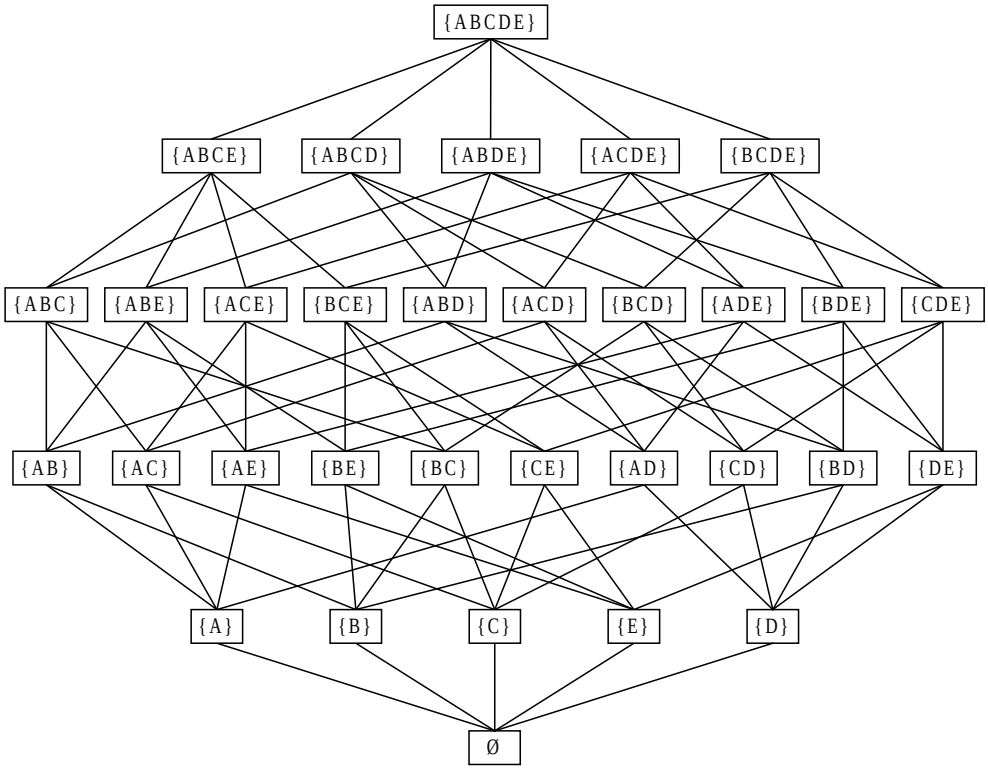


FIG. 1.3 – Treillis des itemsets associé au contexte $\mathcal{D}$.

1.2.3.3 Génération des règles d'association

Étant donnée un ensemble F d'itemsets fréquents dans un contexte d'extraction $\mathcal{B}$ pour un seuil minimal de support *minsupport*, la génération des règles d'association pour un seuil minimal de confiance *minconfiance* est un problème exponentiel dans la taille de F.

Définition 1.3 (Ensemble de règles d'association)
Soit un ensemble F d'itemsets fréquents dans un contexte d'extraction $\mathcal{B}$ pour un seuil minimal de support minsupport. *Étant donné un seuil minimal de confiance* minconfiance, *l'ensemble $\mathcal{AR}$ des règles d'association valides dans $\mathcal{B}$ est :*

$$\mathcal{AR} = \{r : l_2 \rightarrow (l_1 - l_2) \quad | \quad l_1, l_2 \in F \ \wedge \ l_2 \subset l_1 \ \wedge \ support(l_1)/support(l_2) \geq \text{minconfiance}\}.$$

En pratique, la génération des règles d'association est réalisée de manière directe, sans accéder au contexte d'extraction, et le coût de cette phase en temps d'exécution

est donc faible comparé au coût de l'extraction des itemsets fréquents. Le principe général de la génération des règles d'association est le suivant.

Pour chaque itemset fréquent l_1 dans F, tous les sous-ensembles l_2 de l_1 sont déterminés et la valeur du rapport $support(l_1)/support(l_2)$ est calculée. Si cette valeur est supérieure ou égale au seuil de confiance *minconfiance* alors la règle d'association $l_2 \rightarrow (l_1 - l_2)$ est générée. Un algorithme efficace de génération des règles d'association a été proposé par Agrawal et al dans [AS94]. Cet algorithme, qui utilise les propriétés des itemsets fréquents afin de réduire le nombre de tests réalisés, est présenté en détail dans la section 3.2.

1.2.3.4 Visualisation et interprétation des résultats

Cette phase consiste en la visualisation par l'utilisateur des règles d'association extraites du contexte et leur interprétation afin d'en déduire des connaissances utiles pour l'amélioration de l'activité concernée. Le nombre important de règles d'association extraites impose le développement d'outils de classification des règles selon leurs propriétés, de sélection de sous-ensembles de règles selon des critères définis par l'utilisateur, et de visualisation de ces règles sous une forme intelligible. Cette forme peut être textuelle, graphique ou bien combiner ces deux formes de représentation. Le développement de tels outils est complexe et important pour le développement de systèmes commerciaux d'extraction de règles d'association [Man97]. Plusieurs travaux concernant le problème de la visualisation de règles d'implication ont été réalisés dans de nombreux domaines. Toutefois, du fait des caractéristiques propres aux règles d'association, telles que le support, la confiance et la nécessité de tenir compte des besoins et des connaissances de l'utilisateur, il est nécessaire de développer des outils spécifiques de visualisation de ces règles [KK96, KMR$^+$94, LHWC99]. Des systèmes d'exploration, de classification et de sélection de sous-ensembles de l'ensemble des règles d'association extraites, ainsi que de visualisation de ces règles ont été proposés. Le système présenté dans [KMR$^+$94], appelé Rule Visualizer, permet à l'utilisateur de sélectionner des règles parmi l'ensemble des règles extraites en définissant des templates (voir section 3.4.1) et de visualiser ces règles sous forme textuelle ou bien sous forme de graphes dirigés. Dans [LHWC99], les règles sont regroupées et classifiées selon leurs divergences par rapport aux connaissances définies par l'utilisateur dans un langage spécifique et leurs mesures de support et de

confiance ; les règles de chaque groupe peuvent ensuite être visualisées sous forme textuelle. Un environnement interactif de visualisation graphique des règles d'association est proposé dans [FYMT96] ; les items sont représentés par des pixels et les mesures de précision des règles par les couleurs qui leurs sont associées.

1.3 Contribution

Durant ce travail, nous nous sommes intéressé à trois problèmes ouverts concernant l'extraction de règles d'association. Ces problèmes, identifiés dans la littérature comme des problème majeurs pour la définition d'algorithmes d'extraction efficaces et pour l'amélioration de la pertinence du résultat, sont les suivants :

- La définition d'une sémantique formelle complète pour le problème de l'extraction de règles d'association [Man97]. Une telle sémantique est nécessaire afin d'identifier les connexions entre le problème de l'extraction de règles d'association et les problèmes autres du KDD et d'autres domaines [GKMT97, FPSS96b]. Elle est également nécessaire pour identifier et analyser les sous-problèmes de l'extraction de règles d'association et définir des propriétés permettant de développer des algorithmes efficaces de résolution de ces sous-problèmes [MT97, FPSS96a].
- Le problème de l'efficacité de l'extraction des règles d'association dans le cas de données denses ou corrélées. Les algorithmes existants ont été développés afin d'extraire les règles à partir de données de ventes de supermarché qui sont éparses et faiblement corrélées [BMUT97, AMS+96]. Cela signifie que le nombre moyen d'items par objets est faible et que chaque item n'est contenu que dans un petit nombre d'objets. Toutefois, les données d'une grande proportion des bases de données réelles sont fortement corrélées ou denses : données statistiques [BMS97, SW85] et spatiales [KH95], collections de textes [SBM98] et d'images [OO98], historiques d'accès Internet [CMS97], etc. Dans ce type de données, le nombre moyen d'items par objet est élevé et de nombreux items sont contenus dans une majorité d'objets. La taille du contexte étant ainsi augmentée, le coût des balayages de ce dernier sont également augmentés. De plus, le nombre d'itemsets fréquents est considérable et leur taille moyenne élevée, ce qui augmente considérablement le nombre d'opérations à réaliser et entraîne des temps de calcul CPU importants. L'application des algorithmes

existants à des données de ce type entraîne des exécutions longues (de plusieurs dizaines de minutes à plusieurs heures) et, du fait du nombre d'applications concernées, l'extraction de règles d'association à partir de ces données est un problème important du KDD [BMUT97, BMS97, BAG99, Uth96].

– Le problème de la pertinence et de l'utilité de l'ensemble de règles d'association générées [HF95, KMR+94, SA95, TKR+95]. Ce problème est lié au nombre de règles d'association extraites qui est très important dans la plupart des cas et aux nombreuses règles redondantes parmi celles-ci qui nuisent à l'interprétation du résultat. Si dans le cas de données faiblement corrélées et éparses le nombre de règles extraites est de l'ordre d'une dizaine de milliers, dans le cas de données corrélées ou denses, ce sont plusieurs dizaines de milliers à plusieurs millions de règles qui sont extraites avec une très forte proportion de règles redondantes. L'utilisateur se trouve alors confronté au problème de la recherche de relations utiles dans cette masse de règles d'association lors de la visualisation et de l'interprétation des résultats. Du fait de la part importante d'applications concernant des jeux de données denses ou corrélées, la génération d'un ensemble de taille réduite de règles maximisant l'information convoyée est un problème important pour la pertinence et l'utilité en pratique des algorithmes d'extraction de règles d'association.

Dans la suite, nous décrivons brièvement nos principales contributions à la résolution de ces trois problèmes.

1.3.1 Nouvelle sémantique pour l'extraction de règles d'association

Nous définissons une nouvelle sémantique basée sur la connexion de Galois pour le problème de l'extraction de règles d'association. Utilisant les opérateurs de fermeture de la connexion de Galois, nous définissons les *itemsets fermés fréquents*, qui forment le *treillis des itemsets fermés fréquents*, et nous démontrons qu'ils constituent un *ensemble générateur* pour les itemsets fréquents et les règles d'association. Cela signifie que les règles d'association peuvent être générées sans accéder à la base de données lorsque les itemsets fermés fréquents sont déterminés. Nous proposons une nouvelle décomposition du problème en trois sous-problèmes qui sont : la découverte des itemsets fermés fréquents, la dérivation des itemsets fréquents à partir

des itemsets fermés fréquents et la génération des règles d'association à partir des itemsets fréquents. Cette décomposition permet de réduire considérablement l'espace de recherche car si le problème de la découverte des itemsets fermés fréquents a une complexité exponentielle dans le pire des cas, des résultats théoriques et expérimentaux ont démontrés que la complexité en moyenne est bien inférieure dans le cas de bases de données réelles [GM94, GMA95]. Lorsque les itemsets fermés fréquents sont déterminés, les itemsets fréquents et les règles d'association peuvent être générés directement, sans accéder au jeu de données.

1.3.2 Algorithmes de découverte des ensembles fermés fréquents

Nous proposons deux nouveaux algorithmes, nommés Close et A-Close, basés sur les opérateurs de fermeture de la connexion de Galois, permettant la détermination efficace des ensembles fermés, selon cette fermeture, dans les (grandes) bases de données. Comparés aux algorithmes existants de détermination des ensembles fermés, ces algorithmes permettent de réduire le nombre d'accès au jeu de données (accès disque) qui constituent les opérations les plus coûteuses en temps lorsque ces algorithmes sont appliqués à de grands volumes de données [TPBL99, Tao00]. De plus, Close et A-Close prennent en compte la notion de support des ensembles fermés, et permettent donc de déterminer les itemsets fermés fréquents, à partir desquels les itemsets fréquents et les règles d'association sont dérivés de manière directe, sans accéder au jeu de données. Comparés aux algorithmes d'extraction de règles d'association basés sur la recherche des itemsets fréquents, ces algorithmes permettent :

- La réduction des temps de calcul CPU et de l'espace mémoire nécessaires à la recherche des itemsets fréquents, plus particulièrement dans le cas de données denses ou corrélées. L'ensemble d'itemsets fermés fréquents étant un sous-ensemble de l'ensemble des itemsets fréquents, le nombre d'opérations nécessaires à leur extraction est inférieur au nombre d'opérations nécessaires à la recherche des itemsets fréquents. De plus, la dérivation des itemsets fréquents à partir des itemsets fermés fréquents est directe et en conséquence, l'espace mémoire et les temps de calcul CPU nécessaires sont considérablement réduits par rapport aux algorithmes basés sur la recherche des itemsets fréquents.

– La réduction du nombre de balayages de la base de données réalisés dans le cas de données denses ou fortement corrélées. Les processus de découverte des itemsets fréquents et des itemsets fermés fréquents sont itératifs, un balayage de la base de données étant réalisé lors de chaque itération. Toutefois, le nombre d'itérations nécessaires à la détermination des itemsets fermés fréquents est inférieur au nombre d'itérations nécessaires à la détermination des itemsets fréquents.

Nous présentons également les résultats de nombreuses expérimentations, réalisées sur des bases de données réelles, qui démontrent que ces algorithmes permettent de réduire le temps de l'extraction des règles d'association et l'espace mémoire utilisé lors de cette extraction.

1.3.3 Algorithmes de génération de bases pour les règles d'association

Le problème de la pertinence et de l'utilité des règles d'association extraites est un des problèmes pratiques majeurs de l'extraction des règles d'association. L'augmentation des seuils minimaux de support et de confiance, afin de diminuer le nombre de règles générées, ne constitue pas une solution satisfaisante car elle entraîne le risque de supprimer des règles utiles car significatives et, de plus, ne permet pas de supprimer les règles redondantes. Nous adaptons des bases, également appelées couvertures réduites, pour les règles d'implication développées dans le domaine de l'analyse de données et nous définissons de nouvelles bases pour les règles d'association. Nous proposons également des algorithmes d'extraction de ces bases pour les règles d'association à partir des itemsets fermés fréquents. Ces bases sont des ensembles de tailles réduites qui minimisent le nombre de règles d'association générées tout en maximisant la quantité et la qualité des informations convoyées. Elles permettent :

– L'élimination des règles d'association redondantes. Les méthodes existantes d'élimination des règles redondantes se basent sur l'utilisation de la taxonomie des attributs pour produire des règles d'association généralisées [HF95, SA95], sur les règles d'inférence [TKR+95] ou sur d'autres mesures statistiques [AY98, BMS97, PS91, SBMU98, SBM98] pour éliminer certaines redondances. Toutefois, aucune de ces méthodes ne permet d'éliminer la totalité des règles

redondantes et le calcul de mesures statistiques nécessite des temps d'exécution importants.
- La présentation à l'utilisateur d'un ensemble de règles couvrant l'ensemble des attributs de la base de données. Ces bases contiennent des règles dont l'union des items constituant les conséquences des règles est égale à l'union des items constituant les conséquences de toutes les règles d'association valides dans le contexte. Il est nécessaire de ne pas limiter la recherche à un seul sous-ensemble des attributs de la base de données car les règles « surprenantes » pour l'utilisateur constituent des informations utiles qu'il est nécessaire de considérer [DL98, Hec96, PSM94, ST95, ST96].
- Les nouvelles bases que nous définissons permettent de plus la génération des règles d'association non redondantes les plus informatives seulement, c'est à dire des règles les plus utiles et pertinentes : celles qui ont un antécédent (partie gauche) minimal et une conséquence (partie droite) maximale.

Dans [BAG99, NLHP98, SVA97], les auteurs proposent de restreindre la recherche à un sous-ensemble d'attributs afin de réduire le nombre de règles générées, toutefois cette solution ne permet pas d'éliminer les redondances et ne fournit qu'un résultat partiel. Nous présentons également les résultats des expérimentations que nous avons réalisé et qui montrent l'efficacité et l'utilité de la génération de ces bases.

1.4 Organisation du mémoire

Ce mémoire est organisé en deux parties. La première partie fait l'état de l'art de l'extraction de règles d'association. Les algorithmes d'extraction des itemsets fréquents sont présentés dans le chapitre 2. Les algorithmes de génération d'ensembles de règles d'association à partir des itemsets fréquents sont décrits dans le chapitre 3. La seconde partie décrit nos contributions à l'amélioration du champ d'application, de l'efficacité et de la qualité du résultat de l'extraction de règles d'association. Dans le chapitre 4, nous présentons une nouvelle sémantique pour l'extraction de règles d'association basé sur la fermeture de la connexion de Galois. Les nouveaux algorithmes d'extraction des itemsets fermés fréquents que nous proposons sont présentés dans le chapitre 5. Dans le chapitre 6, nous exposons de nouveaux algorithmes de génération de bases pour les règles d'association à partir des itemsets fermés fréquents. La conclusion du mémoire et les perspectives de travaux ultérieurs sont présentés

dans le chapitre 7.

Première partie

État de l'art

Chapitre 2

Découverte des ensembles fréquents d'items

Sommaire

2.1 Introduction

Nous commençons par présenter quelques définitions concernant les ensembles ordonnés afin d'introduire le *treillis des parties* associé à l'ensemble des items $\mathcal{I}$, également appelé *treillis des itemsets*. Nous utilisons ensuite ce treillis afin de présenter les diverses méthodes d'extraction des itemsets fréquents.

2.1.1 Rappels sur les ensembles ordonnés

Nous rappelons dans la suite quelques définitions concernant les ensembles ordonnés, les treillis complets et les opérateurs de fermetures. Ces définitions sont extraites des ouvrages de référence [BM70, Bir67, DP94].

Ensembles ordonnés Soit E un ensemble. Un ordre ou ordre partiel sur l'ensemble E est une relation binaire $\leq$ sur les éléments de E telle que pour tout $x, y, z \in E$ nous avons les propriétés suivantes :

1. Réflexivité : $x \leq x$,
2. Anti-symétrie : $x \leq y$ et $y \leq x \Rightarrow x = y$,
3. Transitivité : $x \leq y$ et $y \leq z \Rightarrow x \leq z$.

Un ensemble E doté d'une relation binaire d'ordre $\leq$, noté $(E, \leq)$, est appelé *ensemble ordonné*.

Relation de couverture Soit $(E, \leq)$ un ensemble ordonné et $x, y \in E$ deux éléments de E. La relation de couverture entre les éléments de E, notée $\prec$, est définie par : $x \prec y$ si et seulement si $x \leq y$ et il n'existe pas d'élément $z \in E$ tel que $x \leq z \leq y$ pour $z \neq x$ et $z \neq y$. Si $x \prec y$, nous disons que y *couvre* x ou bien que y est un *successeur immédiat* de x.

Join et Meet Soit un sous-ensemble $S \subseteq E$ de l'ensemble ordonné $(E, \leq)$. Un élément $u \in E$ est un majorant, ou upper-bound, de S si pour tout élément $s \in S$ nous avons $s \leq u$. L'ensemble des majorants de S est noté $UB(S)$. Dualement, un

élément $v \in E$ est un minorant, ou lower-bound, de S si pour tout élément $s \in S$ nous avons $v \leq s$. L'ensemble des minorants de S est noté $LB(S)$.

$$UB(S) = \{u \in S \mid \forall s \in S \text{ nous avons } s \leq u\}$$
$$LB(S) = \{v \in S \mid \forall s \in S \text{ nous avons } v \leq s\}$$

Le plus petit majorant d'un ensemble S, s'il existe, est le plus petit élément de l'ensemble $UB(S)$ des majorants de S. Cet élément est noté $Join$(S). Dualement, le plus grand des minorants d'un ensemble S, s'il existe, est le plus grand élément de l'ensemble $LB(S)$ des minorants de S. Cet élément est noté $Meet$(S).

$$Join(S) = Min_{\leq}(UB(S))$$
$$Meet(S) = Max_{\leq}(UB(S))$$

Treillis complet Un ensemble ordonné $(E, \leq)$ non vide est un *treillis* si pour tout couple d'éléments $x, y \in E$ l'ensemble $\{x, y\}$ possèdent possède un plus petit majorant noté $Join(\{x, y\})$ et un plus grand minorant noté $Meet(\{x, y\})$. L'ensemble ordonné $(E, \leq)$ est un *treillis complet* si pour tout sous-ensemble $S \subseteq E$ les éléments $Join(S)$ et $Meet(S)$ existent. Les treillis sont fréquement représentés sous forme de diagrammes de Hasse [Wil82], également appelé graphes de couverture, dont les sommets sont les éléments de l'ensemble et les arcs correspondent aux relations de couverture entre les sommets.

Les 2^m itemsets potentiellement fréquents pour un ensemble d'items $\mathcal{I}$ de taille m sont les sous-ensembles de l'itemset $\{\bigcup i \in \mathcal{I}\}$ constitué de tous les items de $\mathcal{I}$. Ces itemsets constituent l'ensemble des parties de l'ensemble d'items $\mathcal{I}$ qui est noté $2^{\mathcal{I}}$. L'ensemble ordonné $\mathcal{L}_{\mathcal{I}} = (2^{\mathcal{I}}, \subseteq)$ forme un treillis complet appelé treillis des parties de $\mathcal{I}$, dont la hauteur est $m + 1$. La découverte des itemsets fréquents consiste à extraire les itemsets du treillis des parties associé à $\mathcal{I}$ dont le support dans le contexte d'extraction est supérieur ou égal au seuil minimal de support *minsupport*. Les itemsets fréquents dans le contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 sont mis en évidence dans le diagramme de Hasse représentant le treillis des itemsets de la figure 2.1. Quinze itemsets sont fréquents, l'itemset $\varnothing$ n'étant pas considéré, et le plus grand des itemsets fréquents est {ABCE} dont la taille est quatre. Dans la suite, nous notons μ la taille (nombre d'items) des plus grands itemsets fréquents.

Les premiers travaux concernant la découverte des itemsets fréquents ont montré la nécessité de définir un ordre total sur les itemsets afin de limiter les temps de

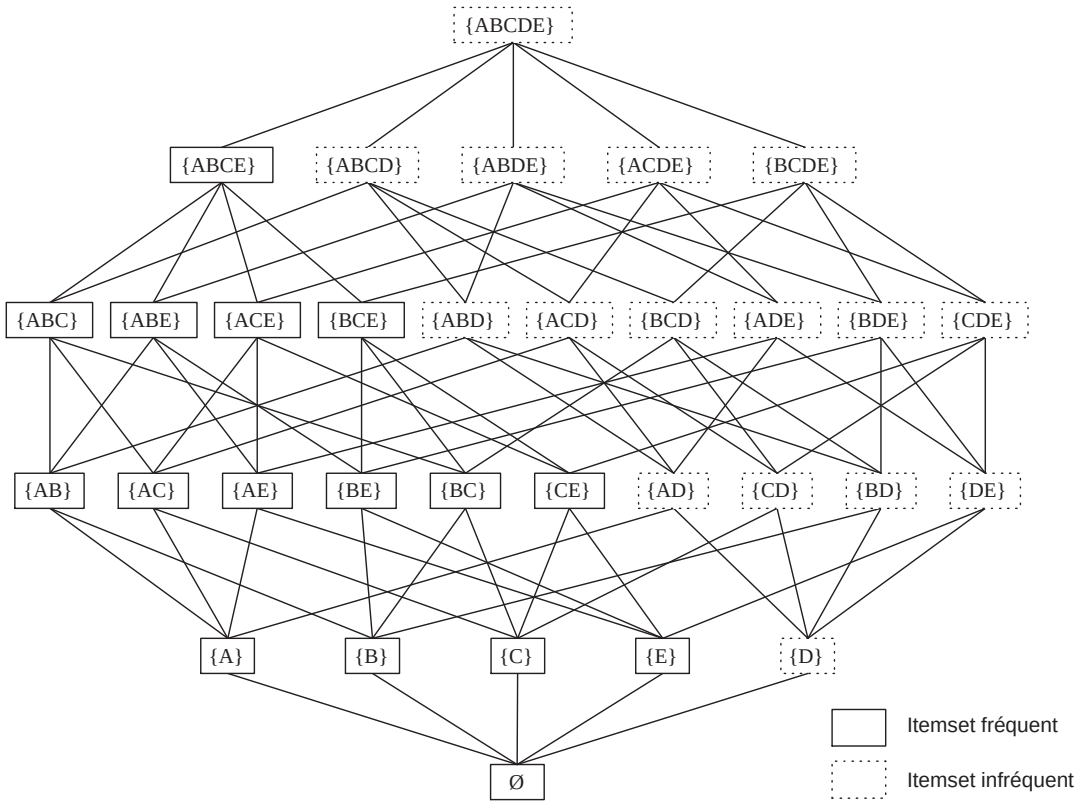


FIG. 2.1 – Itemsets fréquents dans le treillis des itemsets associé au contexte $\mathcal{D}$ pour $minsupport = 2/6$.

réponse. Cet ordre permet d'une part d'éviter les redondances de calcul lors de la recherche, en parcourant chaque itemset du treillis au plus une seule fois, et d'autre part de construire des structures de données autorisant l'exécution efficace des opérations portant sur les itemsets. L'ordre utilisé dans la majorité des travaux publiés est l'*ordre lexicographique*. L'ordre lexicographique, qui est un ordre total sur les itemsets du treillis des parties de $\mathcal{I}$, pour un ensemble de quatre items $\mathcal{I}$ = {A,B,C,D} avec un ordre A < B < C < D sur les items est représenté dans la figure 2.2.

Durant la recherche des itemsets fréquents, des balayages du contexte doivent être réalisés afin de calculer les supports des itemsets et déterminer lesquels sont fréquents. Les opérations d'entrée/sortie liées aux lectures des objets dans le contexte étant des opérations très coûteuses en temps d'exécution, ces balayages représentent une part importante des temps de l'extraction des règles d'association. Le problème

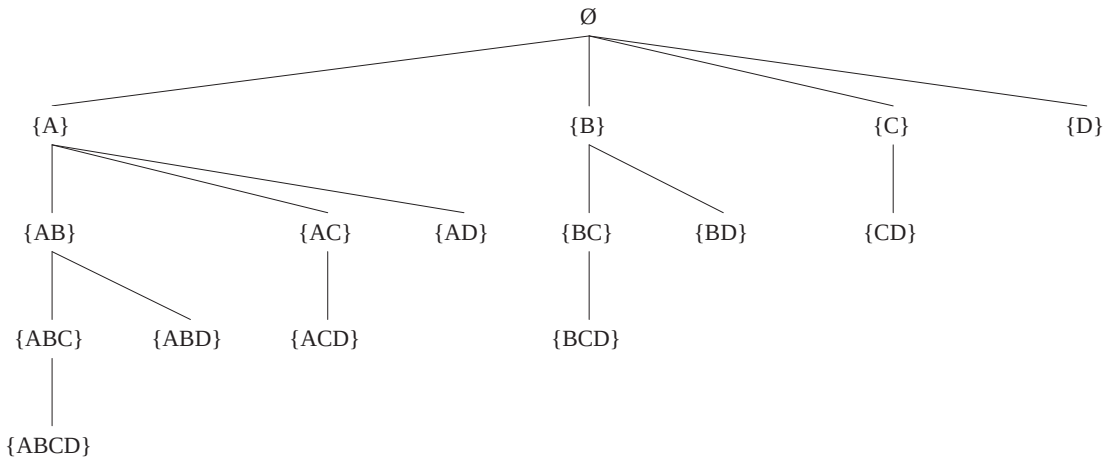


FIG. 2.2 – Ordre lexicographique sur les itemsets pour un ensemble $\mathcal{I}$ de 4 items.

de la réduction du nombre de ces balayages a été l'objet de nombreux travaux de recherche. Deux approches ont été développées afin de résoudre ce problème : l'extraction des itemsets fréquents, présentée dans la section 2.2, et l'extraction des itemsets fréquents maximaux, présentée dans la section 2.3.

2.2 Algorithmes d'extraction des itemsets fréquents

Les algorithmes d'extraction des itemsets fréquents procèdent de manière itérative, en parcourant le treillis des itemsets fréquents « par niveaux » : ils réalisent un parcours en largeur du treillis des itemsets fréquents, du bas vers le haut, en déterminant lors de chaque itération tous les itemsets fréquents d'un niveau, c'est à dire d'une taille donnée. Pour chaque itération k, un ensemble de k-itemsets candidats (k-itemsets fréquents potentiels) est généré et les supports de ces candidats sont calculés lors d'un seul et même balayage, ce qui permet de limiter le nombre total de balayages réalisés. Les premiers algorithmes d'extraction des itemsets fréquents « par niveaux » sont les algorithmes AIS [AIS93b] et SETM [HS95] proposés en 1993.

Plusieurs autres algorithmes permettant de réduire les temps d'extraction des itemsets fréquents ont été proposés. Parmi ceux-ci, nous présentons l'algorithme Apriori [AS94] et l'algorithme OCD [MTV94] basés sur des propriétés et des méthodes identiques. Plusieurs optimisations et structures de données permettant d'améliorer l'efficacité de ces deux algorithmes ont été proposées. Parmi celles-ci, on

peut citer l'algorithme DHP (Direct Hashing and Pruning) proposé par Park et al. [PCY95] qui utilise des tables de hachage afin de diminuer le nombre de candidats générés, l'utilisation de « bitmaps » hiérarchiques proposée par Gardarin et al. [GPW98] qui permet de diminuer le coût des opérations sur les ensembles d'itemsets, la parallélisation du calcul étudiée par Zaki [Zak98] et l'utilisation de heuristiques non-déterministes proposée par Gunopulos et al. [GMS97].

Nous présentons également les algorithmes AprioriTid [AS94], Partition [SON95], Sampling [Toi96b] et DIC (Dynamic Itemset Counting) [BMUT97] qui proposent de nouvelles techniques pour l'extraction des itemsets fréquents. L'algorithme AprioriTid, après le premier balayage du contexte, utilise des listes d'itemsets associées aux identifiants des objets en relation avec les itemsets afin de diminuer progressivement la taille du contexte. Dans l'algorithme Partition, le contexte est décomposé en partitions qui tiennent en mémoire. Pour chaque partition, tous les itemsets fréquents dans la partition sont déterminés puis fusionnés et un balayage de la totalité du contexte est réalisé pour calculer leurs supports sur l'ensemble du contexte. L'algorithme Sampling utilise les techniques d'échantillonnage pour extraire les itemsets fréquents dans un échantillon du contexte et vérifier leurs supports en réalisant un balayage de l'ensemble du contexte. L'algorithme DIC est basé sur une méthode similaire à Apriori. Le contexte est décomposé en sous-contextes d'une taille donnée et durant chaque itération, un sous-contexte (sous-ensemble d'objets) est lu. Ainsi, plusieurs ensembles d'itemsets candidats de tailles différentes sont traités simultanément durant chaque itération, ce qui permet de diminuer le nombre total de balayages du contexte.

2.2.1 Apriori et OCD

L'algorithme Apriori de Agrawal et Srikant [AS94, Sri96] et l'algorithme OCD de Mannila et al. [MTV94] ont été proposés indépendamment en 1994. Ces algorithmes, qui procèdent de la même manière, sont des algorithmes itératifs de recherche des itemsets fréquents par niveaux. Cela signifie que durant la $k^{ème}$ itération, un ensemble d'itemsets candidats de taille k est généré et un balayage du contexte est réalisé afin de supprimer les candidats infréquents. L'ensemble des k-itemsets fréquents ainsi générés est utilisé lors de l'itération $k+1$ suivante pour générer les candidats de taille $k+1$. Ces algorithmes réalisent donc μ itérations afin de déter-

miner tous les itemsets fréquents dans l'ordre croissant de leur tailles, μ étant la taille des plus grands itemsets fréquents. Nous présentons dans la suite l'algorithme Apriori ainsi qu'un exemple d'application de cet algorithme au contexte $\mathcal{D}$.

Afin de limiter le nombre de candidats considérés lors de chaque itération, en réduisant dynamiquement l'espace de recherche des itemsets fréquents, l'algorithme Apriori se base sur les deux propriétés suivantes.

Propriété 2.1 *Tous les sous-ensembles d'un itemset fréquent sont fréquents.*

Cette propriété permet de limiter le nombre de candidats de taille k générés lors de la $k^{ème}$ itération (parmi les 2^k itemsets de taille k existants) en réalisant une jointure conditionnelle des itemsets fréquents de taille $k-1$ découverts lors de l'itération précédente.

Propriété 2.2 *Tous les sur-ensembles d'un itemset infréquent sont infréquents.*

Cette propriété permet de supprimer un candidat de taille k lorsque au moins un de ses sous-ensembles de taille $k-1$ ne fait pas partie des itemsets fréquents découverts lors de l'itération précédente. Le pseudo-code de l'algorithme est présenté dans l'algorithme 2.1. Les notations utilisées sont présentées dans la table 2.1.

C_k	Ensemble de k-itemsets candidats (itemsets potentiellement fréquents). Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
F_k	Ensemble de k-itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

TAB. 2.1 – Notations utilisées dans l'algorithme Apriori.

Durant la première itération de l'algorithme (ligne 1), tous les itemsets de taille 1 sont considérés et un balayage du contexte est réalisé afin de déterminer l'ensemble des 1-itemsets fréquents F_1. Chaque itération k suivante (lignes 2 à 9) se subdivise en deux phases. Durant la première phase (ligne 3), l'ensemble C_k des k-itemsets candidats est construit en joignant les (k-1)-itemsets fréquents dans F_{k-1}. Cette phase est réalisée par la procédure Apriori-Gen. Durant la deuxième phase (lignes 4 à 8), un balayage du contexte est réalisé afin de déterminer le support de chacun des k-itemsets candidats dans C_k et les k-itemsets fréquents sont insérés dans l'ensemble

ALG. 2.1 Extraction des itemsets fréquents avec Apriori.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ;
Sortie : ensembles F_k des k-itemsets fréquents ;

1) $F_1 \leftarrow \{1\text{-itemsets fréquents}\}$;
2) **pour** ($k \leftarrow 2$; $F_{k-1} \neq \varnothing$; k++) **faire**
3) $\quad C_k \leftarrow$ Apriori-Gen(F_{k-1}) ;
4) $\quad$ **pour chaque** objet $o \in \mathcal{B}$ **faire**
5) $\quad\quad C_o \leftarrow$ Subset(C_k, o) ;
6) $\quad\quad$ **pour chaque** candidat $c \in C_o$ **faire** $c.support$++ ;
7) $\quad$ **fin pour**
8) $\quad F_k \leftarrow \{c \in C_k \mid c.support \geq minsupport\}$;
9) **fin pour**
10) **retourner** $\bigcup_k F_k$;

F_k. Lors de ce balayage, pour chaque objet o du contexte, l'ensemble C_o des k-itemsets candidats qui sont contenus dans o est déterminé (ligne 5) et le support de chacun de ces itemsets est incrémenté (ligne 6). La découverte des itemsets contenus dans un objet est réalisée par la procédure Subset. Ces itérations cessent lorsque aucun nouvel itemset candidat ne peut être généré, c'est à dire $F_{k-1} = \varnothing$.

Procédure Apriori-Gen(F_{k-1}) La procédure Apriori-Gen reçoit un ensemble F_{k-1} de (k-1)-itemsets fréquents comme paramètre. Elle retourne un ensemble C_k de k-itemsets candidats qui est un sur-ensemble de l'ensemble des k-itemsets fréquents. Le pseudo-code de la procédure est présenté dans l'algorithme 2.2.

Durant la première partie de la procédure (lignes 1 à 4), deux (k-1)-itemsets fréquents p et q de F_{k-1} sont joints si les $k-2$ premiers items qui les composent sont identiques[1]. Le résultat de cette union est un k-itemset fréquent potentiel (candidat) qui est inséré dans C_k. Durant la deuxième partie (lignes 5 à 9), les k-itemsets candidats dont l'un des sous-ensembles s de taille $k-1$ ne se trouve pas dans F_{k-1} sont supprimés de C_k. Afin d'améliorer l'efficacité de la recherche des sous-ensembles dans F_{k-1}, les itemsets fréquents sont stockés dans une table de hachage.

[1] Le $k^{ème}$ item d'un itemset c est noté $c[k]$.

ALG. 2.2 Génération des itemsets candidats avec Apriori-Gen.

Entrée : ensemble F_{k-1} de (k-1)-itemsets fréquents ;
Sortie : ensemble C_k de k-itemsets candidats ;

1) **insert into** C_k
2) **select** $p[1],\ p[2],\ \ldots,\ p[k-1],\ q[k-1]$
3) **from** $F_{k-1}\ p,\ F_{k-1}\ q$
4) **where** $p[1] = q[1],\ \ldots,\ p[k-2] = q[k-2],\ p[k-1] < q[k-1]$;
5) **pour chaque** itemset candidat $c \in C_k$ **faire**
6) **pour chaque** sous-ensemble s de c de taille $k-1$ **faire**
7) **si** ($s \notin F_{k-1}$) **alors supprimer** c de C_k ;
8) **fin pour**
9) **fin pour**
10) **Retourner** C_k ;

Procédure Subset(C_k, o) La procédure Subset reçoit un ensemble C_k de k-itemsets candidats et un itemset o (nous identifions l'objet o à l'ensemble d'items en relation avec o) comme paramètres. Elle retourne un ensemble C_o de k-itemsets candidats de C_k qui sont contenus dans o. Afin d'améliorer l'efficacité de cette procédure, les itemsets candidats sont stockés dans un *arbre de hachage*. Un noeud de cet arbre contient soit une liste d'itemsets (noeuds feuilles), soit une table de hachage (noeuds intérieurs). Chaque case d'une table de hachage au $k^{ème}$ niveau de l'arbre pointe sur un noeud du $k+1^{ème}$ niveau de l'arbre, la racine de l'arbre se trouvant au niveau 1. Lors de l'ajout d'un itemset dans l'arbre, un parcours en profondeur est réalisé, en partant de la racine de l'arbre et jusqu'à ce qu'une feuille soit atteinte. Quand un noeud intérieur du $j^{ème}$ niveau de l'arbre est atteint, le noeud suivant à parcourir est déterminé en appliquant une fonction de hachage au $j^{ème}$ item de l'itemset. Afin d'équilibrer l'arbre, lorsque le nombre d'itemsets stockés dans un noeud feuille dépasse une seuil fixé, ce noeud est transformé en noeud intérieur. Un exemple d'arbre de hachage est représenté dans la figure 2.3.

Étant donné l'objet o, les itemsets de l'arbre contenus dans o sont déterminés comme suit. Dans le cas ou un noeud feuille de l'arbre est atteint, des pointeurs vers les itemsets stockés dans ce noeud, qui sont contenus dans o, sont ajoutées au résultat. Dans le cas ou un noeud intérieur de l'arbre est atteint (par application de

la fonction de hachage à l'item k de o), alors la fonction de hachage est appliquée à chacun des items venant après k dans o et ce processus est répété récursivement pour chaque noeud résultat. Dans le cas du noeud racine de l'arbre, la fonction de hachage est appliquée à chaque item de o.

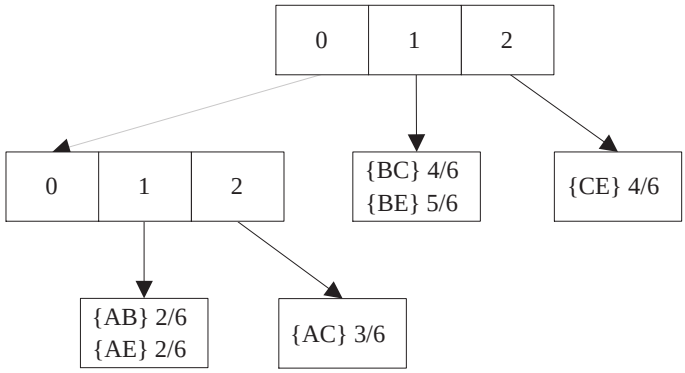


FIG. 2.3 – Exemple d'arbre de hachage.

Exemple 2.1 L'application de l'algorithme Apriori au contexte $\mathcal{D}$ pour un seuil minimal de support *minsupport* de 2/6 est représentée dans la figure 2.4. Quatre itérations sont exécutées par l'algorithme qui calcule quatre ensembles de candidats et d'itemsets fréquents et réalise quatre balayages du contexte. Cette exécution est identique à l'exécution de l'algorithme OCD pour ce même contexte et pour le même seuil minimal de support.

2.2.2 AprioriTid

L'algorithme AprioriTid est une variante de l'algorithme Apriori proposée par Agrawal et Srikant dans [AS94]. Il permet de diminuer progressivement la taille du contexte, dans le but de le stocker en mémoire et ne plus réaliser d'opération d'entrée/sortie, après le premier balayage de celui-ci. Durant les itérations suivant la première, la procédure Apriori-Gen est utilisée pour générer les itemsets candidats de l'itération suivante dans un ensemble $\overline{C}_k$. Chaque élément de l'ensemble $\overline{C}_k$ est un couple (OID,$\{c_k\}$) dans lequel $\{c_k\}$ est la liste des k-itemsets candidats contenus dans l'objet dont l'identifiant est OID. Le support d'un itemset candidat c_k correspond donc au nombre d'apparitions de c_k dans l'ensemble $\overline{C}_k$. Le pseudo-

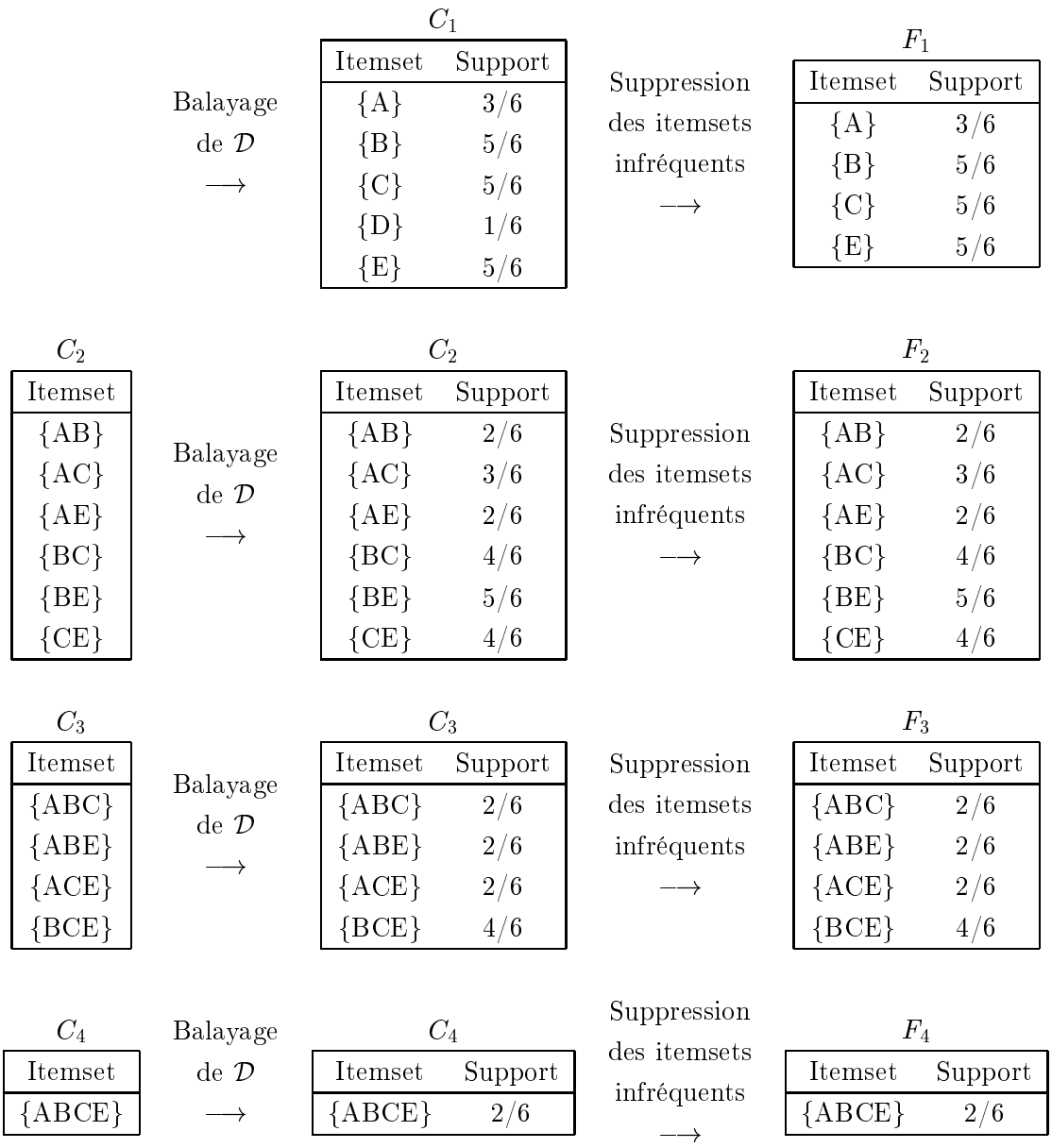


FIG. 2.4 – Extraction des itemsets fréquents dans le contexte $\mathcal{D}$ avec Apriori pour $minsupport = 2/6$.

code de l'algorithme est présenté dans l'algorithme 2.3. Les notations utilisées sont présentées dans la table 2.2.

$\overline{C}_k$	Chaque élément de cet ensemble correspond à un objet et possède deux champs : *OID* qui est l'identifiant de l'objet et *liste-candidats* qui contient la liste des k-itemsets candidats contenus dans l'objet.
C_k	Ensemble de k-itemsets candidats (itemsets potentiellement fréquents). Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
F_k	Ensemble de k-itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

TAB. 2.2 – Notations utilisées dans l'algorithme AprioriTid.

L'ensemble $\overline{C}_1$ correspond au contexte après transformation de chaque item i en itemset $\{i\}$ (ligne 2). Pour chaque $k^{ème}$ itération suivante ($1 < k \leq \mu$), l'ensemble C_k est généré en utilisant la procédure Apriori-Gen (ligne 4) et l'ensemble $\overline{C}_k$ est construit en utilisant les ensembles $\overline{C}_{k-1}$ et C_k (ligne 6 à 10). Chaque élément de $\overline{C}_k$ correspond à un objet o de $\overline{C}_{k-1}$ et contient son OID et la liste des k-itemsets candidats de C_k contenus dans o. L'ensemble F_k est construit en déterminant pour chaque candidat de C_k son nombre d'apparition dans $\overline{C}_k$ et en insérant dans F_k les candidats fréquents (ligne 11).

Structures de données À chaque itemset candidat est associé un identifiant unique, appelé *ID*, créé par incrémentation d'un compteur lors de la création du candidat. Chaque ensemble C_k est stocké dans un tableau indexé par les *ID* des k-itemsets candidats qu'il contient. Chaque ensemble $\overline{C}_k$ est stocké dans une structure séquentielle ; chaque élément de $\overline{C}_k$ est de la forme (*OID*,{*ID*}). Chaque k-itemset candidat de C_k est créé par la procédure Apriori-Gen en joignant deux (k-1)-itemsets fréquents dans F_{k-1}. Pour chaque k-itemset candidat dans C_k et chaque k-itemset fréquent dans F_k, deux champs supplémentaires sont stockés : *générateurs* et *extensions*. Le champ *générateurs* d'un k-itemset candidat c_k contient les *ID* des (k-1)-itemsets fréquents f^1_{k-1} et f^2_{k-1} joints pour créer c_k. Le champ *extensions* d'un k-itemset fréquent f_k contient les *ID* des $(k+1)$-itemsets candidats c_{k+1} qui sont des extensions de f_k, c'est à dire tels que $c_{k+1} = f_k \cup i$ avec $i \in (\mathcal{I} \setminus f_k)$. Lors de sa création du candidat c_k par jointure des itemsets fréquents f^1_{k-1} et f^2_{k-1}, le *ID* de c_k est ajouté au champs *extensions* de f^1_{k-1} et f^2_{k-1}.

Utilisant les structures décrites précédemment, l'algorithme détermine les candi-

ALG. 2.3 Extraction des itemsets fréquents avec AprioriTid.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ;
Sortie : ensembles F_k des k-itemsets fréquents ;

1) $F_1 \leftarrow \{\text{1-itemsets fréquents}\}$;
2) $\overline{C}_1 \leftarrow \mathcal{B}$;
3) **pour** $(k \leftarrow 2 ; F_{k-1} \neq \varnothing ; k{+}{+})$ **faire**
4) $\quad C_k \leftarrow \text{Apriori-Gen}(F_{k-1})$;
5) $\quad \overline{C}_k \leftarrow \varnothing$;
6) $\quad$ **pour chaque** objet o **tel que** $o.OID \in \overline{C}_{k-1}$ **faire**
$\quad\quad$ // Détermine les candidats de C_k contenus dans o
7) $\quad\quad C_o \leftarrow \{c \in C_k \mid (c - c_k) \in o.\textit{liste-candidats} \wedge (c - c_{k-1}) \in o.\textit{liste-candidats}\}$;
8) $\quad\quad$ **pour chaque** candidat $c \in C_o$ **faire** $c.\textit{support}{+}{+}$;
9) $\quad\quad$ **si** $(C_o \neq \varnothing)$ **alors** $\overline{C}_k \leftarrow \overline{C}_k \cup \{(o.OID, C_k)\}$;
10) $\quad$ **fin pour**
11) $\quad F_k \leftarrow \{c \in C_k \mid c.\textit{support} \geq \textit{minsupport}\}$;
12) **fin pour**
13) **retourner** $\bigcup_k F_k$;

dats de C_k contenus dans un objet o (ligne 7) comme suit. Le champ *liste-candidats* d'un objet o dans l'ensemble $\overline{C}_{k-1}$ contient la liste des *ID* des (k-1)-itemsets candidats contenus dans o. Le champ *extensions* de chacun de ces (k-1)-itemsets candidats c_{k-1} donne la liste des k-itemsets qui sont des extensions de c_{k-1}. Pour chacun des k-itemsets c_k extension de c_{k-1}, le champ *générateurs* donne les *ID* des deux itemsets joints pour créer c_k. Si les *ID* de ces deux itemsets sont présents dans l'ensemble *o.liste-candidats* alors c_k est également contenu dans o et est inséré dans C_o. La recherche d'un *ID* dans l'ensemble *o.liste-candidats* est réalisée en temps constant en utilisant une table binaire. Chaque entrée dans la table correspond à un *ID* de C_k. La table est créée au début du parcours de $\overline{C}_k$ et est initialisée pour chaque objet o de $\overline{C}_k$.

Exemple 2.2 L'application de l'algorithme AprioriTid au contexte $\mathcal{D}$ pour un seuil minimal de support *minsupport* de 2/6 est représentée dans la figure 2.5.

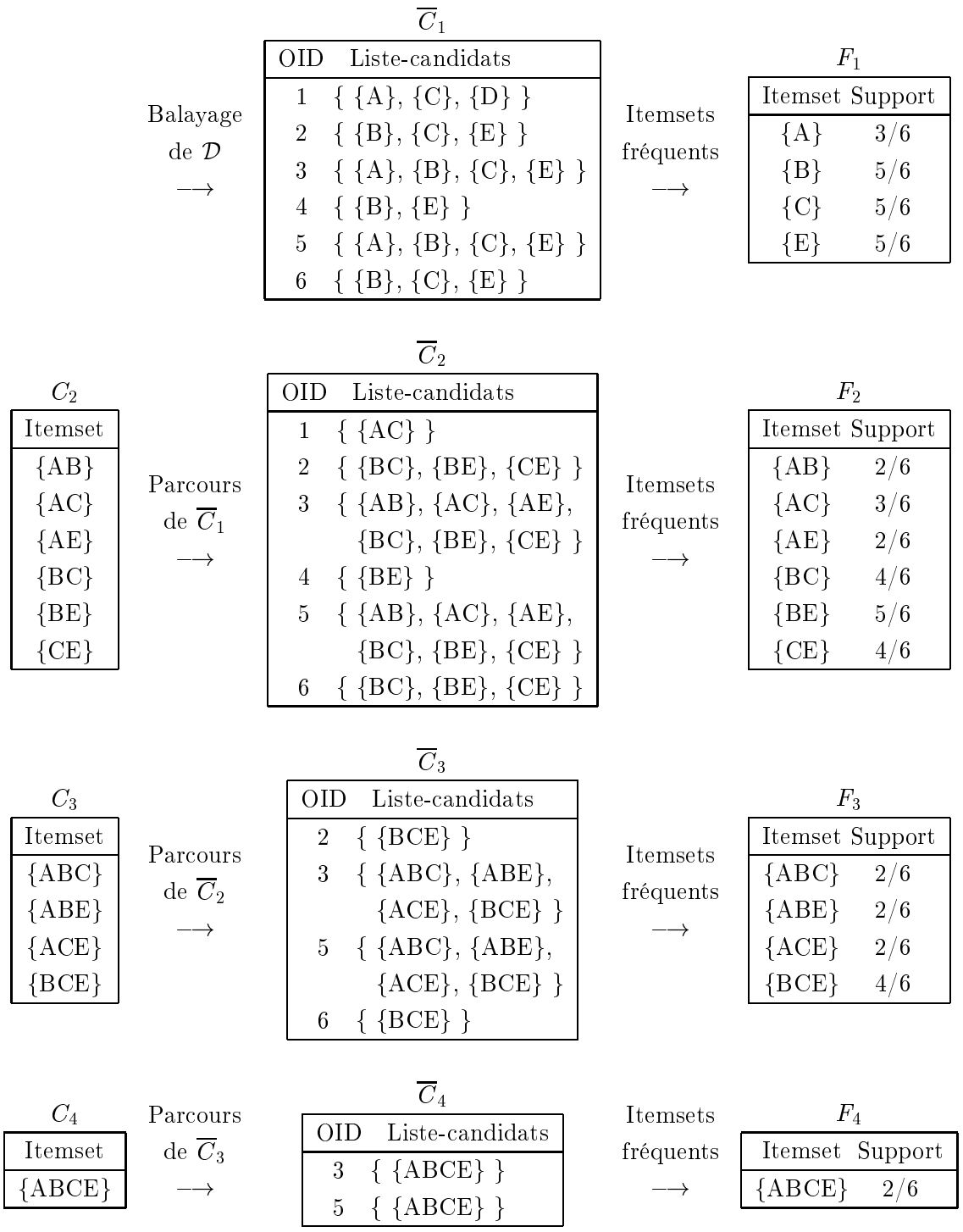


FIG. 2.5 – Extraction des itemsets fréquents dans le contexte $\mathcal{D}$ avec AprioriTid pour $minsupport = 2/6$.

2.2.3 Partition

L'algorithme Partition a été proposé par Savasere et al. [SON95] en 1995. Il permet de ne réaliser que deux balayages du contexte afin d'extraire les itemsets fréquents. Durant le premier balayage, le contexte est divisé en n partitions qui sont considérées une à une successivement. Pour chaque partition, l'ensemble des itemsets fréquents dans la partition (itemsets fréquents locaux) sont extraits. Les ensembles d'itemsets fréquents dans chaque partition sont ensuite fusionnés afin d'obtenir un ensemble d'itemsets candidats potentiellement fréquents sur l'ensemble de la relation binaire (itemsets candidats globaux). L'ensemble ainsi obtenu est un sur-ensemble de l'ensemble des itemsets fréquents. Durant le second balayage, les supports de ces itemsets candidats sont calculés et les itemsets fréquents sur l'ensemble du contexte (itemsets fréquents globaux) sont identifiés. La taille des partitions est choisie de telle manière que l'ensemble des objets d'une partition tienne en mémoire. Le pseudo-code de l'algorithme est présenté dans l'algorithme 2.4. Les notations utilisées sont présentées dans la table 2.3.

n	Nombre de partitions du contexte.
P_r	$r^{ème}$ partition du contexte.
C_k^G	Ensemble de k-itemsets candidats globaux (itemsets fréquents potentiels). Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
C^G	Ensemble d'itemsets candidats globaux (itemsets fréquents potentiels). Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
F^r	Ensemble d'itemsets fréquents dans la partition P_r. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
F^G	Ensemble d'itemsets fréquents globaux (itemsets fréquents). Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

TAB. 2.3 – Notations utilisées dans l'algorithme Partition.

L'algorithme procède en quatre phases. Durant la première phase (ligne 1), le contexte est divisée en n partitions disjointes. Durant la seconde phase (lignes 2 à 5), chaque partition P_r pour $1 \leq r \leq n$ est lue et la procédure Partition-Gen est appliquée afin d'en extraire l'ensemble F^r des itemsets fréquents locaux. Les ensembles F^r $(1 \leq r \leq n)$ ainsi obtenus sont ensuite fusionnés lors de la troisième phase (lignes 6 à 8) afin d'obtenir l'ensemble C^G des candidats globaux. Chaque ensemble C_k^G des

candidats globaux de taille k est construit par l'union des ensembles F_k^r contenant les k-itemsets fréquents dans la partition P_r (ligne 7). Durant la quatrième phase (lignes 9 à 13), les n partitions sont lues successivement et les supports des itemsets candidats de C^G sur l'ensemble de la relation (support globaux) sont déterminés (lignes 9 à 12). L'ensemble F^G des itemsets fréquents globaux est construit en identifiant les candidats globaux dont le support global est supérieur ou égal à *minsupport* (ligne 13).

ALG. 2.4 Extraction des itemsets fréquents avec Partition.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ; nombre de partitions n ;

Sortie : ensembles F^G des itemsets fréquents ;

// Création des partitions P_1 à P_n
1) Partitionner($\mathcal{B}$,n) ;
// Extraction des itemsets fréquents dans chaque partition
2) **pour** ($r \leftarrow 1$; $r \leq n$; r++) **faire**
3) **lire** partition P_r ;
4) $F^r \leftarrow$ Partition-Gen(P_r, *minsupport*) ;
5) **fin pour**
// Fusions des itemsets fréquents extraits dans chaque partition
6) **pour** ($k \leftarrow 1$; $F_k^r \neq \varnothing$; k++) **faire**
7) $C_k^G \leftarrow \bigcup_{r=1}^{r=n} F_k^r$;
8) **fin pour**
// Calcul des supports globaux des itemsets
9) **pour** ($r \leftarrow 1$; $r \leq n$; r++) **faire**
10) **lire** partition P_r ;
11) Partition-Count(C^G, P_r) ;
12) **fin pour**
13) $F^G \leftarrow \{c \in C^G \mid c.support \geq minsupport\}$;
14) **retourner** F^G ;

Procédure Partition-Gen(P_r, *minsupport*) La procédure Partition-Gen reçoit une partition P_r du contexte et un seuil minimal de support *minsupport* comme

paramètres. Elle retourne l'ensemble F^r des itemsets fréquents locaux, c'est à dire les itemsets qui sont fréquents dans la partition P_r, quelle que soit leur taille. Le pseudo-code de la procédure est présenté dans l'algorithme 2.5.

ALG. 2.5 Génération des itemsets fréquents locaux avec Partition-Gen.

Entrée : partition P_r du contexte $\mathcal{B}$; seuil minimal de support *minsupport* ;
Sortie : ensemble F^r des itemsets fréquents locaux ;

1) $F_1^{P_r} \leftarrow$ {1-itemsets fréquents dans P_r avec liste de OID} ;
2) **pour** ($k \leftarrow 2$; $F_k^{P_r} \neq \varnothing$; k++) **faire**
3) **pour chaque** itemset $p \in F_{k-1}^{P_r}$ **faire**
4) **pour chaque** itemset $q \in F_{k-1}^{P_r}$ **faire**
5) **si** ($p[1] = q[1] \ \wedge \ \ldots \ \wedge \ p[i-2] = q[i-2] \ \wedge \ p[i-1] < q[i-1]$)
6) **alors faire**
7) $c \leftarrow p[1] \cup p[2] \cup \ldots \cup p[i-1] \cup q[i-1]$;
8) **si** ($\forall s$ sous-ensemble de c de taille $|c|$-1 nous avons $s \in F_{k-1}^{P_r}$)
9) **alors faire**
10) $c.listeOID \leftarrow p.listeOID \cap q.listeOID$;
11) **si** ($|c.listeOID|/|P_r| \geq minsupport$) **alors**
12) $F_k^{P_r} \leftarrow F_k^{P_r} \cup \{c, c.support\}$;
13) **finsi**
14) **finsi**
15) **fin pour**
16) **fin pour**
17) **fin pour**
18) **Retourner** $\bigcup F_k^{P_r}$;

À chaque itemset est associée un champ *listeOID* contenant la liste des OID des objets de la partition qui contiennent l'itemset. Après la lecture de la partition P_r, l'ensemble $F_1^{P_r}$ des 1-itemsets fréquents est construit (ligne 1). Chaque élément de $F_1^{P_r}$ est un couple $(f_1, \{OID\})$ dans lequel f_1 est un 1-itemset et $\{OID\}$ est la liste des identifiants des objets de la partition contenant f_1. Le support de f_1 dans la partition (support local) correspond donc au rapport entre la taille de la liste $f_1.listeOID$ et le nombre d'objets de la partition. Afin de créer les k-itemsets fréquents pour $k > 1$, les (k-1)-itemsets fréquents sont joints et leurs champs *listeOID* sont utilisés pour

déterminer les supports comme suit.

La première phase de la procédure (lignes 3 à 7) est identique à la première phase de Apriori-Gen. Deux (k-1)-itemsets fréquents p et q de $F_{k-1}^{P_r}$ sont joints si les $k-2$ premiers items qui les composent sont identiques afin de créer un k-itemset candidat c. Durant la deuxième phase (lignes 8 à 12), chaque candidat c créé pendant la première phase est considéré. Si tous les sous-ensembles de c de taille $k-1$ sont présents dans $F_{k-1}^{P_r}$ (ligne 8), le champ *listeOID* de c est initialisée avec l'intersection des champs *listeOID* des itemsets p et q (ligne 10). Cette intersection fournit la liste des OID des objets contenant c. Le support local de c est alors déterminé en divisant la taille de la liste *listeOID* par le nombre d'objets de la partition. Si le support local de c est supérieur à *minsupport* alors c est inséré avec son support local dans $F_k^{P_r}$.

Procédure Partition-Count(C^G, P_r) La procédure Partition-Count reçoit l'ensemble C^G des candidats globaux de toutes les tailles et une partition P_r du contexte comme paramètres. Elle met à jour le support global de chaque candidat c de C^G en fonction du nombre d'objets de la partition P_r contenant c. Le pseudo-code de la procédure est présenté dans l'algorithme 2.6.

ALG. 2.6 Détermination des supports globaux avec Partition-Count.

Entrée : ensemble C^G de candidats globaux ; partition P_r du contexte $\mathcal{B}$;
Sortie : champs *support* des candidats de C^G mis à jour ;

1) **pour chaque** 1-itemset $c_1 \in C_1^G$ **faire**
2) $c_1.listeOID \leftarrow \{o.OID \mid o \in P_r$ et c_1 contenu dans $o\}$;
3) $c_1.support \leftarrow c_1.support + |c_1.listeOID|$;
4) **fin pour**
5) **pour** ($k \leftarrow 2$; $C_k^G \neq \varnothing$; k++) **faire**
6) **pour chaque** k-itemset $c \in C_k^G$ **faire**
7) $templist \leftarrow c[1].listeOID \cap c[2].listeOID \cap \ldots \cap c[k].listeOID$;
8) $c.support \leftarrow c.support + |templist|$;
9) **fin pour**
10) **fin pour**

L'algorithme commence par construire pour chaque 1-itemset c_1 de C^G la liste des OID des objets de la partition contenant c_1 et met à jour le support de c_1 en

fonction de la taille de cette liste (lignes 1 à 4). Ensuite, pour chaque itemset $c \in C^G$ de taille supérieure à un, la procédure détermine la liste *templist* des OID des objets contenant c par intersection des *listeOID* de tous les 1-itemsets inclus dans c (ligne 7). Si par exemple, pour les 1-itemsets fréquents {A} et {B} nous avons {A}.*listeOID* = {1,3} et {B}.*listeOID* = {2,3}, alors pour l'itemset {AB} nous aurons *templist* = {3}. Le support global de l'itemset c est ensuite mis à jour en augmentant sa valeur de la taille de la liste *templist* (ligne 8).

Exemple 2.3 L'application de l'algorithme Partition au contexte $\mathcal{D}$ pour un seuil minimal de support *minsupport* de 2/6 et en divisant $\mathcal{D}$ en deux partitions de trois objets *(n = 2)* est représentée dans la figure 2.6.

Structures de données La génération des itemsets fréquents locaux est réalisée par la procédure Partition-Gen durant la deuxième phase de l'algorithme. Pour chaque itemset c, la liste c.*listeOID* des OID des objets contenant c est stockée sous forme de tableau, chaque élément du tableau étant un OID. Étant donné que les objets apparaissent dans l'ordre croissant de leur OID dans le contexte, les OID sont triés par ordre croissant dans les listes *listeOID* lors de leur création. En conséquence, la liste générée pour un itemset c par intersection des listes des itemsets p et q (utilisés pour créer c) est également triée par ordre croissant des OID et le coût de l'opération d'intersection correspond à un parcours unique de chacune des deux listes.

L'ensemble C^G des candidats globaux est obtenu en fusionnant les ensembles d'itemsets fréquents locaux de chaque partition. C^G est initialisé avec la liste des itemsets fréquents locaux de la première partition et les itemsets fréquents locaux des partitions suivantes sont insérés si C^G ne les contient pas déjà. Afin d'améliorer l'efficacité de cette opération les itemsets candidats globaux sont stockés dans une table de hachage.

Il est possible d'éliminer certains itemsets candidats globaux de C^G, avant de déterminer leurs supports globaux avec la procédure Partition-Count, en utilisant leurs supports locaux. Rappelons que pour un itemset candidat c, le support local de c pour chaque partition dans laquelle il est fréquent est connu. Les itemsets qui peuvent être supprimés de C^G correspondent à deux catégories :

- Les itemsets pour lesquels le support global est déjà connu. Ce sont les itemsets qui sont fréquents dans chacune des partitions. Leurs supports locaux dans

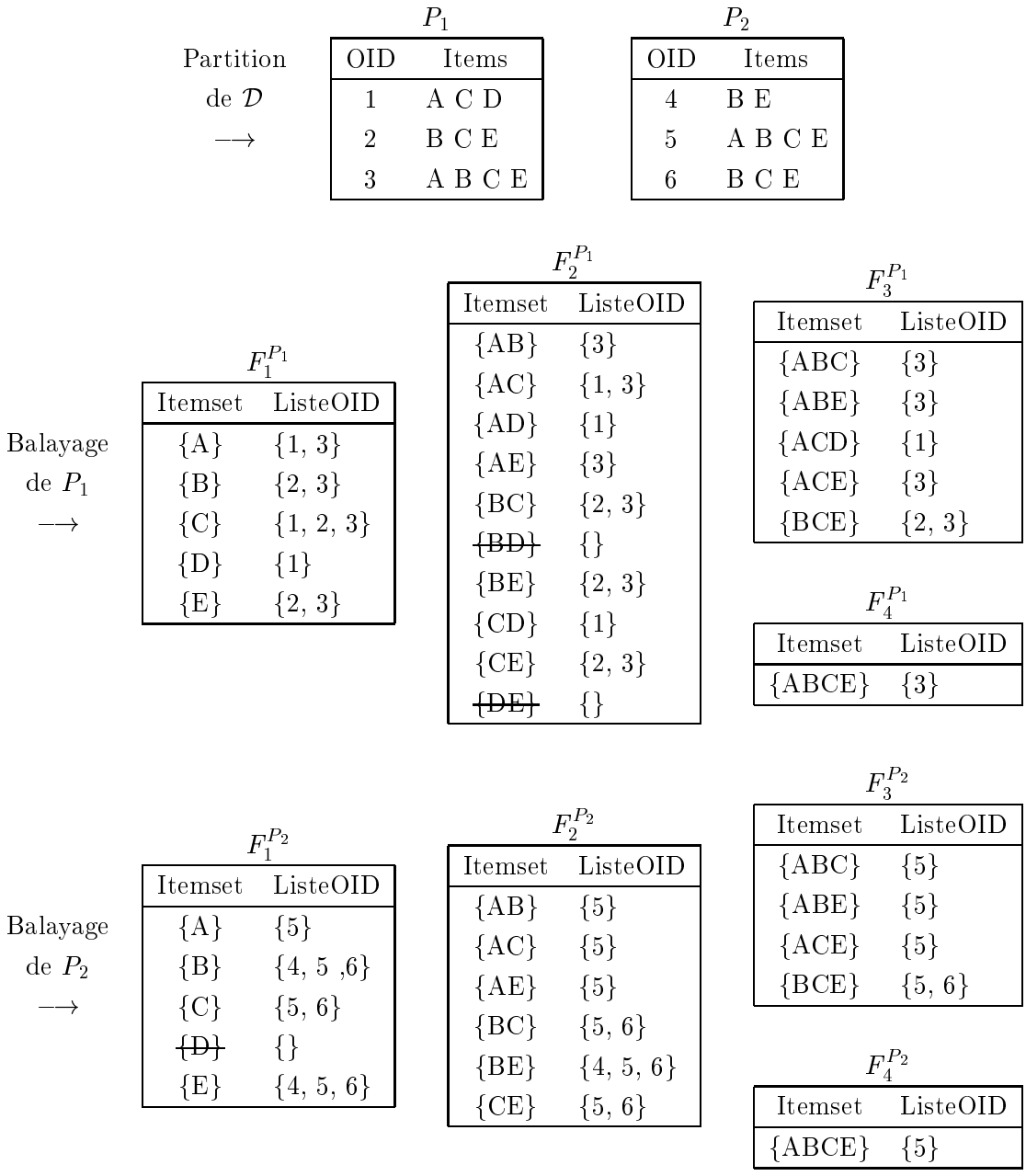


FIG. 2.6 – Extraction des itemsets fréquents dans le contexte $\mathcal{D}$ avec Partition pour $minsupport = 2/6$ et $n = 2$.

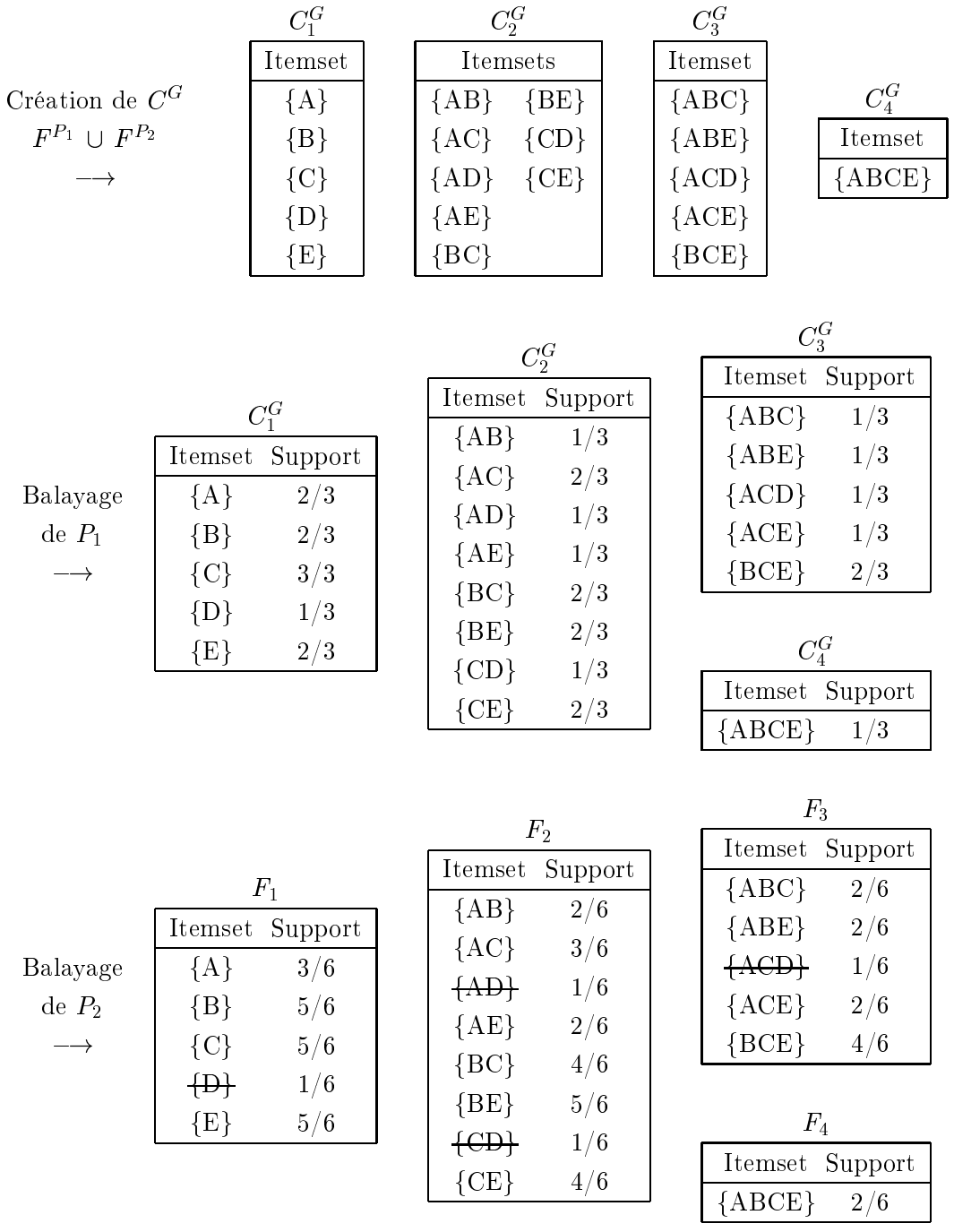


FIG. 2.6 – Extraction des itemsets fréquents dans le contexte $\mathcal{D}$ avec Partition pour *minsupport* = 2/6 et $n = 2$ (*suite et fin*).

chaque partition étant connu, leurs supports globaux peuvent être déterminés directement. Ces itemsets sont supprimés de C^G et sont directement insérés dans l'ensemble des itemsets fréquents globaux.

- Les itemsets pour lesquels le support global ne peut être supérieur à *minsupport*. Un itemset c pour lequel la somme des supports de c pour les partitions dans lesquelles il est fréquent plus le nombre de ces partitions est inférieur à la somme des seuils de support minimaux pour les partitions dans lesquelles c est fréquent plus le nombre de ces partitions ne peut pas être un itemset fréquent global. Cet itemset c ainsi que tous ses sur-ensembles sont supprimés de C^G.

Cette optimisation nécessite le stockage pour chaque itemset candidat global de son support cumulé pour les partitions dans lesquelles il est fréquent et d'un compteur du nombre de ces partitions.

Le calcul des supports globaux des candidats est réalisé en appliquant la procédure Partition-Count à chaque partition du contexte. Les structures de données utilisées lors de cette phase sont identiques à celles utilisées pour la génération des itemsets fréquents locaux. Pour chaque 1-itemset candidat global c_1, la liste $c_1.listeOID$ des OID des objets de la partition lue contenant c_1 est créée et stockée dans un tableau. Le support d'un itemset candidat global c est mis à jour pour cette partition en réalisant l'intersection des *listeOID* de tous les 1-itemsets inclus dans c, en déterminant la taille de la liste résultante et en augmentant $c.support$ de cette valeur.

2.2.4 Sampling

L'utilisation de l'échantillonnage pour la découverte des itemsets fréquents fut introduite par Toivonen [Toi96b, Toi96a] en 1996 avec l'algorithme Sampling. Cet algorithme utilise les notions de *bordure positive* et *bordure négative*. Soit un contexte $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, un seuil minimal de support *minsupport* et l'ensemble F des itemsets fréquents dans $\mathcal{B}$ selon *minsupport*. Les bordures positive et négative de F sont définies comme suit.

Définition 2.1 (Bordure positive)
La bordure positive de F notée $Bd^+(F)$ est constituée des itemsets maximaux inclus dans F :

$$Bd^+(F) = \{f \subseteq \mathcal{I} \mid f \in F \ \wedge \ \forall f' \supset f \text{ nous avons } f' \notin F\}.$$

La bordure positive correspond aux itemsets fréquents maximaux, c'est à dire les itemsets fréquents dont tous les sur-ensembles stricts[2] sont infréquents.

Définition 2.2 (Bordure négative)
La bordure négative de F notée $Bd^-(F)$ est constituée des itemsets minimaux qui ne sont pas inclus dans F :

$$Bd^-(F) = \{f \subseteq \mathcal{I} \mid f \notin F \ \wedge \ \forall f' \subset f \text{ nous avons } f' \in F\}.$$

La bordure négative contient les itemsets infréquents dont tous les sous-ensembles stricts sont fréquents. Ces itemsets sont ceux dont les supports sont les plus proches du seuil minimal de support parmi les itemsets infréquents.

Exemple 2.4 Les bordures positive et négative pour les itemsets fréquents dans le contexte $\mathcal{D}$ pour un seuil minimal de support de 1/6 sont représentés dans la figure 2.7. La bordure positive contient les deux itemsets {ABCE} et {ACD} et la bordure négative contient les deux itemsets {BD} et {DE} qui sont respectivement les itemsets fréquents maximaux et les itemsets infréquents minimaux dans $\mathcal{D}$ pour $minsupport = 1/6$.

Le principe de l'approche consiste à déterminer un échantillon aléatoire du contexte, extraire les itemsets fréquents dans l'échantillon et la bordure négative de ces itemsets fréquents, et calculer ensuite le support réel de ces itemsets fréquents et des itemsets de la bordure négative sur la totalité du contexte. Afin de limiter le risque d'erreur, c'est à dire de diminuer la probabilité de « manquer » un itemset fréquent, le seuil minimal de support utilisé pour l'extraction des itemsets fréquents dans l'échantillon est diminué. Le pseudo-code de l'algorithme est présenté dans l'algorithme 2.7. Les notations utilisées sont présentées dans la table 2.4.

L'algorithme commence par déterminer l'échantillon aléatoire E_s sur le contexte (ligne 1). L'ensemble S des itemsets fréquents dans l'échantillon E_s pour le seuil minimal de support diminué σ est généré par la procédure Sampling-Gen (ligne 2). La bordure négative $Bd^-(S)$ est déterminée (ligne 3) et l'ensemble C des itemsets candidats est construit en fusionnant S et $Bd^-(S)$ (ligne 4). L'ensemble F des itemsets candidats de C qui sont fréquents sur l'ensemble du contexte pour le seuil minimal de support *minsupport* est généré par la procédure Sampling-Count (ligne 5). Si des

[2]Nous appelons sur-ensemble strict d'un itemset l un itemset l' tel que $l \subset l'$.

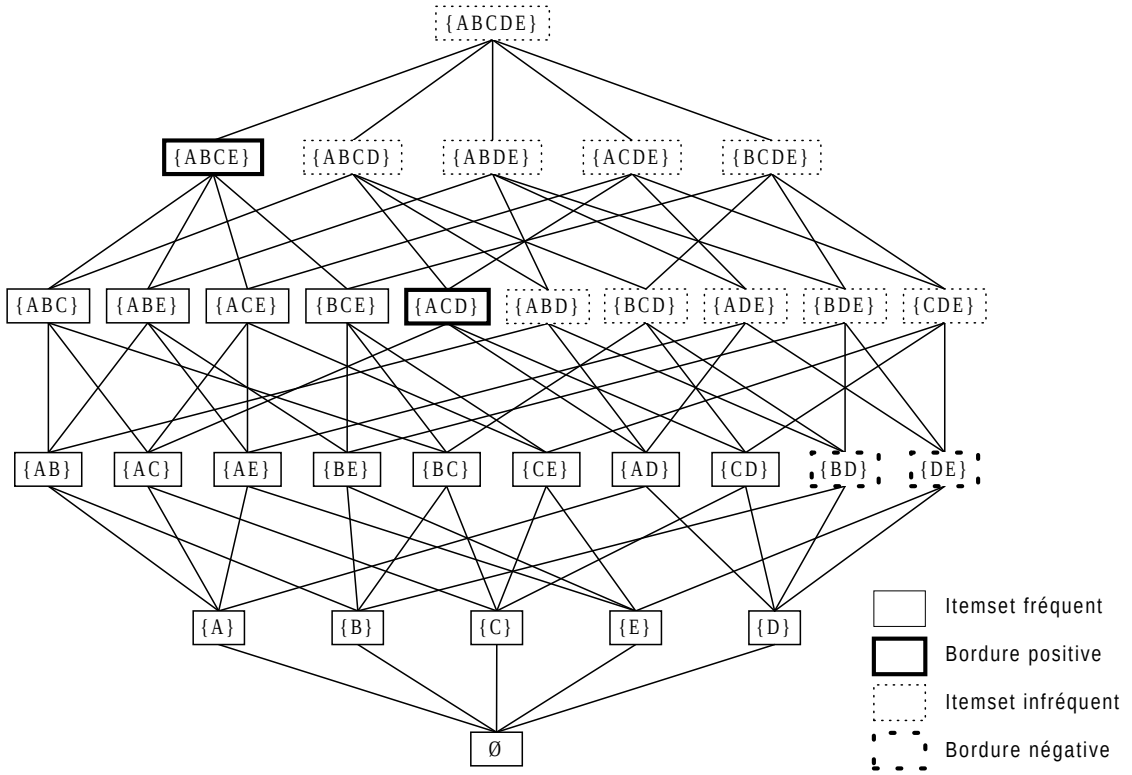


FIG. 2.7 – Bordures positive et négative des itemsets fréquents dans le contexte $\mathcal{D}$ pour $minsupport = 1/6$.

s	Nombre d'objets de l'échantillon du contexte.
E_s	Échantillon du contexte.
σ	Valeur diminuée de *minsupport* afin de limiter le risque d'erreur.
C	Ensemble d'itemsets candidats. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
F	Ensemble d'itemsets fréquents extraits. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

TAB. 2.4 – Notations utilisées dans l'algorithme Sampling.

itemsets de la bordure négative sont des itemsets fréquents alors il est possible que certains itemsets fréquents n'est pas été générés et un message en averti l'utilisateur (ligne 6).

ALG. 2.7 Extraction des itemsets fréquents avec Sampling.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ; seuil minimal de support diminué σ ; taille de l'échantillon s ;

Sortie : ensemble F des itemsets fréquents ou bien un sous-ensemble de F et un rapport d'erreur possible ;

// Création de l'échantillon du contexte
1) $E_s \leftarrow$ échantillon aléatoire de $\mathcal{B}$ de taille s ;
// Recherche des itemsets fréquents dans l'échantillon
2) $S \leftarrow$ Sampling-Gen(E_s, σ) ;
// Recherche des itemsets fréquents dans le contexte
3) **calculer** $Bd^-(S)$;
4) $C \leftarrow S \cup Bd^-(S)$;
5) $F \leftarrow$ Sampling-Count$(\mathcal{B}, C, minsupport)$;
6) **si** $(F \cap Bd^-(S) \neq \varnothing)$ **alors message** "Erreur possible" ;
7) **retourner** F ;

Procédure Sampling-Gen(E_s, σ) La procédure Sampling-Gen génère les ensembles S_k des k-itemsets fréquents dans l'échantillon E_s pour le seuil de support diminué σ. Le pseudo-code de la procédure est présenté dans l'algorithme 2.8.

L'ensemble C_1 des 1-itemsets candidats est initialisé avec la liste des items du contexte (ligne 1) et le compteur d'itérations k est initialisé à 1 (ligne 2). Durant chaque itération k suivante (lignes 3 à 7), l'ensemble S_k des k-itemsets candidats de C_k qui sont fréquents dans E_s pour le seuil minimal de support σ est construit par la procédure Sampling-Count (ligne 4). L'ensemble C_{k+1} des candidats de taille $k+1$ est généré en appliquant la procédure Apriori-Gen à l'ensemble S_k (ligne 5) et le compteur k est incrémenté (ligne 6). Ces itérations cessent lorsque aucun nouveau candidat ne peut être généré et la procédure retourne les ensembles S_k de k-itemsets fréquents dans E_s selon σ (ligne 8).

Procédure Sampling-Count$(O, C,$ *valsupport*$)$ La procédure Sampling-Count détermine les itemsets candidats d'un ensemble C qui sont fréquents dans un ensemble d'objets O pour un seuil minimal de support *valsupport*. Les objets de l'ensemble O sont lus une et une seule fois par la procédure. Le pseudo-code est présenté

ALG. 2.8 Génération des itemsets fréquents de l'échantillon avec Sampling-Gen.

Entrée : échantillon E_s du contexte ; seuil minimal de support diminué σ ;

Sortie : ensembles S_k des k-itemsets fréquents dans E_s selon σ ;

1) $C_1 \leftarrow \{\{i\} \mid i \in \mathcal{I}\}$;
2) $k \leftarrow 1$;
3) **tant que** $(C_k \neq \varnothing)$ **faire**
4) $\quad S_k \leftarrow$ Sampling-Count(E_s, C_k, σ) ;
$\quad$ // Génération des candidats de taille $k+1$
5) $\quad C_{k+1} \leftarrow$ Apriori-Gen(S_k) ;
6) $\quad k{+}{+}$;
7) **fin tant que**
8) **retourner** $\bigcup_k S_k$;

dans l'algorithme 2.9.

A chaque item $i \in \mathcal{I}$ du contexte est associé un champ *contenu_dans* dans lequel sont stockés les références aux itemsets candidats de C qui contiennent i. A chaque itemset candidat $c \in C$ est associé un champ *count* correspondant au nombre d'objets de O contenant l'itemset c. Les champs $i.contenu\_dans$ et $c.count$ sont initialisés durant la phase d'initialisations de la procédure (lignes 1 à 7). Pour chaque itemset $c \in C$, un compteur $c.item\_count$ est utilisé afin de déterminer les objets $o \in O$ contenant c (lignes 8 à 16). Pour chaque objet $o \in O$ lu, les compteur $c.item\_count$ sont initialisés à zéro (ligne 9) et les items contenus dans l'objet o sont considérés un à un (lignes 10 à 15). Pour chaque item $i \in o$, les compteurs des candidats $c \in C$ contenant i sont incrémentés (lignes 11 et 12) et si la valeur du compteur est égale à la taille de c alors tous les items de c sont contenus dans o (c est donc contenu dans o) et la valeur $c.count$ est incrémentée (ligne 13). La procédure retourne tous les candidats dont le support dans O est supérieur ou égal au seuil minimal $valsupport$ (ligne 17).

2.2.5 Dynamic Itemset Counting

L'algorithme Dynamic Itemset Counting (DIC) a été proposé par Brin et al. [BMUT97] en 1997. Il permet d'améliorer l'efficacité de la recherche des itemsets fréquents par niveaux en diminuant le nombre de balayages du contexte réalisés. Le

ALG. 2.9 Détermination des candidats fréquents avec Sampling-Count.

Entrée : ensemble d'objets O ; ensemble d'itemsets candidats C ; valeur minimale de support *valsupport* ;

Sortie : ensemble des itemsets candidats fréquents dans O selon *valsupport* ;

// Initialisations
1) **pour chaque** item $i \in \mathcal{I}$ **faire** $i.contenu\_dans \leftarrow \varnothing$;
2) **pour chaque** candidat $c \in C$ **faire**
3) **pour chaque** item $i \in c$ **faire**
4) $i.contenu\_dans \leftarrow i.contenu\_dans \cup \{c\}$;
5) **fin pour**
6) **fin pour**
7) **pour chaque** candidat $c \in C$ **faire** $c.count \leftarrow 0$;
// Lectures de l'ensemble d'objets O
8) **pour chaque** objet $o \in O$ **faire**
9) **pour chaque** candidat $c \in C$ **faire** $c.item\_count \leftarrow 0$;
10) **pour chaque** item $i \in o$ **faire**
11) **pour chaque** candidat $c \in i.contenu\_dans$ **faire**
12) $c.item\_count{+}{+}$;
13) **si** $(c.item\_count = |c|)$ **alors** $c.count{+}{+}$;
14) **fin pour**
15) **fin pour**
16) **fin pour**
17) **retourner** $\bigcup\{c, c.support \mid c \in C \ \wedge \ c.count/|O| \geq valsupport\}$;

principe de l'approche consiste à définir une fenêtre M correspondant à un nombre d'objets du contexte et effectuer les lectures du contexte par blocs de M objets. Lorsque la fin du contexte est atteinte, un balayage complet a été effectué et les lectures reprennent au début du contexte. Après la lecture de chaque bloc, les supports des itemsets candidats sont mis à jour, de nouveaux itemsets candidats sont crées et les itemsets pour lesquels tous les objets du contexte ont été parcourus sont soit insérés dans l'ensemble des itemsets fréquents, soit marqués comme itemsets infréquents selon leurs supports.

Le pseudo-code de l'algorithme est présenté dans l'algorithme 2.10. Les notations

utilisées sont présentées dans la table 2.5. L'algorithme utilise un compteur *nombrelu* et un marqueur *situation* pour chaque itemset candidat. Le compteur *nombrelu* indique le nombre d'objets du contexte qui ont été considérés jusque-là pour déterminer le support de l'itemset. Lorsque $nombrelu = |\mathcal{B}|$, tous les objets du contexte ont été considérés et le support de l'itemset est définitif. La valeur du marqueur *situation* dépend d'une part si le support actuel de l'itemset à été calculé sur l'ensemble du contexte ou sur une partie seulement et d'autre par si cet itemset est fréquent ou non selon cette valeur du support. Les valeurs du marqueur *situation* correspondants aux quatre états possibles de l'itemset sont :

M	Taille de la fenêtre : nombre d'objets lus avant de mettre à jour les supports des itemsets candidats.
C_k	Ensemble de k-itemsets candidats. Chaque élément de cet ensemble possède quatre champs : *itemset*, *support*, *nombrelus* et *situation*.
C	Ensemble de tous les itemsets candidats des ensembles C_k quelle que soit leur taille.
P	Ensemble des itemsets candidats des ensembles C_k dont le *support* a été modifié après la lecture de M objets.
F_k	Ensemble de k-itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

TAB. 2.5 – Notations utilisées dans l'algorithme DIC.

- *FC* (Fréquent Confirmé) pour un itemset dont le support calculé sur la totalité du contexte est supérieur à *minsupport* : l'itemset est un itemset fréquent.
- *IC* (Infréquent Confirmé) pour un itemset dont le support calculé sur la totalité du contexte est inférieur à *minsupport* : l'itemset est un itemset infréquent.
- *FP* (Fréquent Potentiel) pour un itemset dont le support n'est pas entièrement calculé, dans le cas ou sa valeur actuelle est supérieure à *minsupport*.
- *IP* (Infréquent Potentiel) pour un itemset dont le support n'est pas entièrement calculé, dans le cas ou sa valeur actuelle est inférieure à *minsupport*.

L'algorithme Apriori correspond à l'application de cette méthode pour une fenêtre de la taille du contexte $M = |\mathcal{B}|$: les 1-itemsets sont déterminés fréquents ou infréquents confirmés à la fin de la première lecture d'un bloc (contexte entier), les 2-itemsets à la fin de la deuxième lecture d'un bloc, etc. En conséquence, il n'est pas

nécessaire pour Apriori d'associer un compteur *nombrelu* et un marqueur *situation* aux itemsets candidats.

L'initialisation de l'algorithme (ligne 1) consiste à insérer tous les 1-itemsets dans l'ensemble C_1 des candidats de taille 1 et initialiser leur marqueur *situation* à *IP*. Lors de la création d'un candidat dans un ensemble C_k les compteurs *support* et *nombrelu* sont initialisés à 0. Durant la première phase de l'algorithme (lignes 2 à 8), M objets du contexte sont lus. Pour chaque objet o lu, le support des candidats qui sont contenus dans o pour lesquels la base n'a pas été entièrement parcourue est incrémenté et ces candidats sont insérés dans l'ensemble P. Lors de la seconde phase qui concerne la création de nouveaux candidats, les itemsets candidats c dans P dont le support est devenu supérieur ou égal à *minsupport* sont considérés successivement (lignes 9 à 15). Le marqueur *situation* de c est positionné à *FP* et on détermine parmi tous ses sur-ensembles x de taille $|c|+1$ ceux dont tous les sous-ensembles de taille $|x|-1$ sont marqués *FP* ou *FC*. Tous les sous-ensembles de x étant au moins temporairement fréquents, x est inséré dans l'ensemble $C_{|x|}$ avec le marqueur *situation* initialisé à *IP*. Durant la troisième phase (lignes 16 à 22), les compteurs du nombre d'objets considérés pour chaque candidat sont mis à jour et les candidats pour lesquels un balayage complet du contexte a été réalisé sont marqués *FC* si leur support est supérieur ou égal à *minsupport* et marqués *IC* dans le cas contraire. Ces itemsets ne seront donc plus considérés lors de la deuxième phase. Pendant la quatrième phase (lignes 23 à 25), si des candidats dans C possèdent des marqueurs *FP* ou bien *IP*, il est nécessaire de continuer le processus et l'algorithme reprend à la première phase (ligne 2). Dans le cas contraire, tous les itemsets fréquents ont été déterminés et les ensembles F_k de k-itemsets fréquents sont construits en insérant dans F_k les candidats de l'ensemble C_k dont le marqueur est *FC*.

Structures de données La structure de stockage des itemsets est identique à celle utilisée par l'algorithme Apriori avec deux informations supplémentaires associées aux itemsets :

- Un marqueur *situation* indiquant si l'itemset est fréquent confirmé, infréquent confirmé, fréquent potentiel ou infréquent potentiel.
- Un marqueur indiquant à quel endroit dans le contexte le calcul du support de l'itemset a commencé. Ce marqueur remplis les fonctions du compteur *nombrelu* utilisé dans l'algorithme.

ALG. 2.10 Extraction des itemsets fréquents avec DIC.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ; taille de la fenêtre de lecture M ;

Sortie : ensembles F_k des k-itemsets fréquents ;

1) $C_1 \leftarrow \{1\text{-itemsets avec } situation = IP\}$;
   // Lecture d'un bloc et mise à jour des supports
2) **lire** M objets dans $\mathcal{B}$;
3) **pour chaque** objet o lu **faire**
4) **pour chaque** candidat $c \subseteq o$ **tel que** $c.situation = IP$ ou FP **faire**
5) $c.support++$;
6) **insérer** c dans P ;
7) **fin pour**
8) **fin pour**
   // Création de nouveaux candidats
9) **pour chaque** candidat $c \in P$ **tel que** $c.situation = IP$ et $c.support \geq minsupport$ **faire**
10) $c.situation \leftarrow FP$;
11) **pour chaque** sur-ensemble x de c **tel que** $|x| = |c| + 1$ et $x \notin C_{|x|}$ **faire**
12) **si** ($\forall s$ sous-ensemble de x de taille $|x| - 1$ nous avons $s.situation =$
13) FC ou FP) **alors insérer** x dans $C_{|x|}$ avec $x.situation = IP$;
14) **fin pour**
15) **fin pour**
   // Mise à jour des compteurs d'objets testés
16) **pour chaque** candidat $c \in C$ **tel que** $c.situation = IP$ ou FP **faire**
17) $c.nombrelu \leftarrow c.nombrelu + M$;
18) **si** ($c.nombrelu = |\mathcal{B}|$) **alors faire**
19) **si** ($c.support \geq minsupport$) **alors** $c.situation \leftarrow FC$;
20) **sinon** $c.situation \leftarrow IC$;
21) **finsi**
22) **fin pour**
   // Test d'arrêt
23) **si** ($\exists c \in C \mid c.situation = FP$ ou IP) **alors aller** ligne 2 ;
24) $F_k \leftarrow \{c \in C_k \mid c.situation = FC\}$;
25) **retourner** $\bigcup_k F_k$;

Le problème de la mise à jour des supports après la lecture d'un objet dans le contexte (lignes 5 à 8) est un problème important pour l'efficacité de l'algorithme. L'algorithme DIC réalise un moins grand nombre de lectures d'objets que Apriori, mais pour chaque objet il doit mettre à jour le support d'un plus grand nombre de candidats. Prenons par exemple le cas d'une fenêtre M correspondant au tiers de la taille du contexte. Pendant la première lecture d'un bloc, ce sont les supports des 1-itemsets doivent être mis à jour pour chaque objet lu, pendant la deuxième ce sont les supports des 1 et 2-itemsets, pendant la troisième ceux des 1, 2 et 3-itemsets, pendant la quatrième ceux des 2, 3 et 4-itemsets, etc.

Afin de réduire le coût de ces mises à jour, Brin et al. ont proposé d'ordonner les items par ordre croissant de fréquence d'apparition dans le contexte. En effet, étant donné un ensemble d'itemsets, la structure de l'arbre de hachage dépend fortement de l'ordre des items. Un représentation schématique de l'arbre de hachage, en utilisant l'ordre lexicographique, pour les itemsets {ABC}, {AD}, {BC} et tous leurs sous-ensembles est donnée dans la figure 2.8.

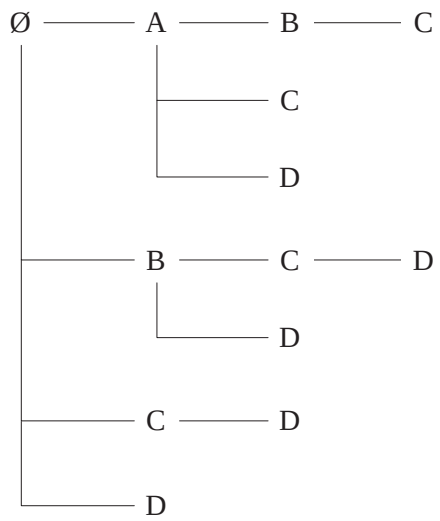


FIG. 2.8 – Représentation schématique d'un arbre de hachage utilisant l'ordre lexicographique.

On peut observer que l'item A apparaît une seule fois dans l'arbre alors que l'item D apparaît cinq fois. Le coût de l'incrémentation des supports des itemsets

contenus dans un objet o est :

$$\sum_{R} n - index(dernier(R), o)$$

ou R correspond aux itemsets noeuds intérieurs de l'arbre qui sont contenus dans o et $n - index(dernier(R), o)$ est le nombre d'items restant dans o après le dernier item de R. Afin de diminuer le coût de cette incrémentation, il est donc préférable que les items apparaissant dans de nombreux objets soient les derniers de l'ordre considérés et les items apparaissant dans peu d'objets soient les premiers. La méthode proposée dans DIC consiste à ordonner les items par ordre croissant de leurs supports après la lecture du premier bloc d'objets. Ceci permet d'obtenir une bonne approximation de l'ordre optimal des items tout en limitant le coût de cet ordonnancement puisque durant la lecture du premier bloc d'objets seuls les 1-itemsets sont considérés et donc aucune arborescence entre les itemsets n'est encore construite. Après cette lecture, l'ordre des items est modifié et l'arbre est construit à ce moment là.

Exemple 2.5 L'application de l'algorithme DIC au contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 et une fenêtre de taille M de 3 objets est représentée dans la figure 2.9. L'algorithme DIC réalise la lecture de 5 blocs d'objets, ce qui représente 2,5 balayages du contexte. Toutefois, la contrepartie de cette diminution du nombre balayages est l'augmentation du nombre de candidats dont le support doit être calculé après chaque lecture. L'ensemble C_2 des 2-itemsets candidats contient 10 itemsets pour DIC alors que Apriori ne génère que 6 itemsets candidats de taille 2. Ceci est dû au fait que l'itemset {D} est un 1-itemset fréquent potentiel après la lecture du premier bloc et est donc utilisé pour construire les 2-itemsets candidats alors que {D} est infréquent sur la totalité du contexte.

C_1

Itemset	Support	Situation
{A}	0/0	*IP*
{B}	0/0	*IP*
{C}	0/0	*IP*
{D}	0/0	*IP*
{E}	0/0	*IP*

Lecture des objets 1 à 3 $\longrightarrow$

C_1

Itemset	Support	Situation
{A}	2/3	*FP*
{B}	2/3	*FP*
{C}	3/3	*FP*
{D}	1/3	*FP*
{E}	2/3	*FP*

C_2

Itemset	Support	Situation
{AB}	0/0	*IP*
{AC}	0/0	*IP*
{AD}	0/0	*IP*
{AE}	0/0	*IP*
{BC}	0/0	*IP*
{BD}	0/0	*IP*
{BE}	0/0	*IP*
{CD}	0/0	*IP*
{CE}	0/0	*IP*
{DE}	0/0	*IP*

Lecture des objets 4 à 6 $\longrightarrow$

C_1

Itemset	Support	Situation
{A}	3/6	*FC*
{B}	5/6	*FC*
{C}	5/6	*FC*
{D}	1/6	*IC*
{E}	5/6	*FC*

C_2

Itemset	Support	Situation
{AB}	1/3	*FP*
{AC}	1/3	*FP*
{AD}	0/3	*IP*
{AE}	1/3	*FP*
{BC}	2/3	*FP*
{BD}	0/3	*IP*
{BE}	3/3	*FP*
{CD}	0/3	*IP*
{CE}	2/3	*FP*
{DE}	0/3	*IP*

C_3

Itemset	Support	Situation
{ABC}	0/0	*IP*
{ABE}	0/0	*IP*
{ACE}	0/0	*IP*
{BCE}	0/0	*IP*

FIG. 2.9 – Extraction des itemsets fréquents dans le contexte $\mathcal{D}$ avec DIC pour $minsupport = 2/6$ et $M = 3$.

Lecture des objets 1 à 3 $\longrightarrow$

C_2

Itemset	Support	Situation
{AB}	2/6	*FC*
{AC}	3/6	*FC*
{AD}	1/6	*IC*
{AE}	2/6	*FC*
{BC}	4/6	*FC*
{BD}	0/6	*IC*
{BE}	5/6	*FC*
{CD}	1/6	*IC*
{CE}	4/6	*FC*
{DE}	0/6	*IP*

C_3

Itemset	Support	Situation
{ABC}	1/3	*FP*
{ABE}	1/3	*FP*
{ACE}	1/3	*FP*
{BCE}	2/3	*FP*

C_4

Itemset	Support	Situation
{ABCE}	0/0	*IP*

Lecture des objets 4 à 6 $\longrightarrow$

C_3

Itemset	Support	Situation
{ABC}	2/6	*FP*
{ABE}	2/6	*FP*
{ACE}	2/6	*FP*
{BCE}	4/6	*FP*

C_4

Itemset	Support	Situation
{ABCE}	1/3	*FP*

Lecture des objets 1 à 3 $\longrightarrow$

C_4

Itemset	Support	Situation
{ABCE}	2/6	*FC*

FIG. 2.9 – Extraction des itemsets fréquents dans le contexte $\mathcal{D}$ avec DIC pour *minsupport* = 2/6 et $M = 3$ (*suite et fin*).

2.2.6 Discussion

Les deux facteurs principaux de l'efficacité, et donc des temps de réponses, des algorithmes d'extraction des itemsets fréquents sont le nombre de balayages du jeu de données réalisés et le nombre d'itemsets candidats considérés par l'algorithme. L'importance du nombre de balayages réalisés est liée au coût des opérations d'entrée/sortie. L'importance du nombre d'itemsets candidats vient du fait que les opérations portant sur ces derniers constituent la majeure partie du temps de calcul CPU de l'algorithme.

Les algorithmes présentés dans cette section ont été développés pour des applications concernant des bases de données commerciales. Les jeux de données utilisés pour leur expérimentation sont construits à partir de bases de données de ventes de supermarchés et de jeux de données synthétiques générés selon les caractéristiques des données de ventes. Ces données sont éparses et faiblement corrélées et les temps d'exécution obtenus sur ces jeux de données sont faibles, de l'ordre de quelques secondes à quelques minutes. Ces temps de réponse sont relativement faibles car dans les données de ce type les plus long itemsets fréquents ne contiennent qu'un nombre limité d'items (la valeur de μ est faible devant m) et le nombre total d'itemsets fréquents (dont le nombre d'itemsets candidats considérés dépend) est réduit. Dans le cas de contextes pour lesquels les plus longs itemsets fréquents sont grands, c'est à dire pour une valeur de μ élevée, les performances de ces algorithmes se dégradent considérablement :

- Apriori et OCD réalisent μ itérations et donc μ balayages du contexte pour extraire les itemsets fréquents. Pour des valeurs élevées de μ, ceci entraîne des temps d'exécutions importants, les opérations d'entrée/sortie étant très coûteuses en temps. Ce problème est essentiel dans le cas de données denses pour lesquelles la taille des jeux de données est plus importante pour un nombre d'objets et un nombre d'items identiques.
- AprioriTid nécessite durant son exécution le stockage des listes d'OID associées aux itemsets dont le volume total peut être plus important que la taille du contexte d'extraction. Ce problème d'espace de stockage nécessite l'utilisation du disque et entraîne des temps d'exécutions importants, particulièrement lors des premières itérations.
- Pour Partition et Sampling, le nombre de k-itemsets candidats pour $1 \leq k \leq$

μ est plus important que pour les algorithmes Apriori et OCD. Ceci entraîne un problème d'espace mémoire nécessaire au stockage de tous les candidats de toutes tailles lors de la phase finale de l'algorithme. De plus, le dernier balayage du contexte qui permet de calculer les supports de tous les itemsets candidats nécessite un temps de calcul important dont une part est inutile car elle concerne des itemsets infréquents globalement.

- Pour DIC, après la $\frac{|\mathcal{B}|}{M}^{ème}$ itération, $\frac{|\mathcal{B}|}{M}$ ensembles d'itemsets candidats (de tailles différentes) sont considérés simultanément lors de chaque itération. De plus, chacun de ces ensembles C_k de k-itemsets candidats est un sur-ensemble de l'ensemble C_k généré par Apriori et OCD. Se posent alors les problèmes de l'espace de stockage des ensembles d'itemsets candidats traités simultanément lors des lectures du jeu de données et du coût total du calcul des supports des candidats qui est plus important que pour Apriori ou ODC. En effet, DIC détermine les supports de certains itemsets candidats infréquents qui ne sont pas considérés par Apriori et OCD.

Les bases de données caractérisées par une valeur de μ élevée représentent une part importante des bases de données réelles. Il est donc nécessaire de développer de nouveaux algorithmes d'extraction des itemsets fréquents qui rendent possible l'extraction de règles d'association à partir de ce type de données dans des temps raisonnables. Les données denses ou corrélées sont également caractérisées par une valeur de μ élevée. Toutefois, dans ce type de données, le problème de l'efficacité des algorithmes d'extraction des itemsets fréquents est exacerbé par une forte densité des itemsets fréquents de grande taille.

2.3 Algorithmes d'extraction des itemsets fréquents maximaux

Les algorithmes d'extraction des itemsets fréquents maximaux ont été développés afin de réduire les temps d'extraction des itemsets fréquents dans les jeux de données pour lesquels la taille des plus longs itemsets fréquents μ est élevée. L'objectif de ces algorithmes est de réduire l'espace de recherche, et donc le nombre d'itemsets candidats considérés, et de diminuer le nombre de balayages du jeu de données réalisés pendant l'extraction. Pour cela, ils sont basés sur une nouvelle décomposition

du problème de l'extraction des itemsets fréquents fondée sur les propriétés 2.1 et 2.2 (section 2.2.1) et sur la définition 2.3 de l'ensemble des itemsets fréquents maximaux. Un itemset fréquent est maximal si tous ses sur-ensembles sont infréquents.

Définition 2.3 (Ensemble des itemsets fréquents maximaux)
Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$. *Étant donné un seuil minimal de support* minsupport, *l'ensemble* M *des itemsets fréquents maximaux dans* $\mathcal{B}$ *est :*

$M = \{l \subseteq \mathcal{I} \mid \text{support}(l) \geq \text{minsupport} \wedge \forall l' \supset l$ *nous avons* $\text{support}(l') < \text{minsupport}\}$.

Les itemsets fréquents maximaux dans le contexte $\mathcal{D}$ pour un seuil minimal de support de 1/6 sont représentés dans la figure 2.10. Pour un tel support, vingt itemsets du treillis sont fréquents et deux itemsets fréquents sont maximaux : {ABCE} et {ACD}.

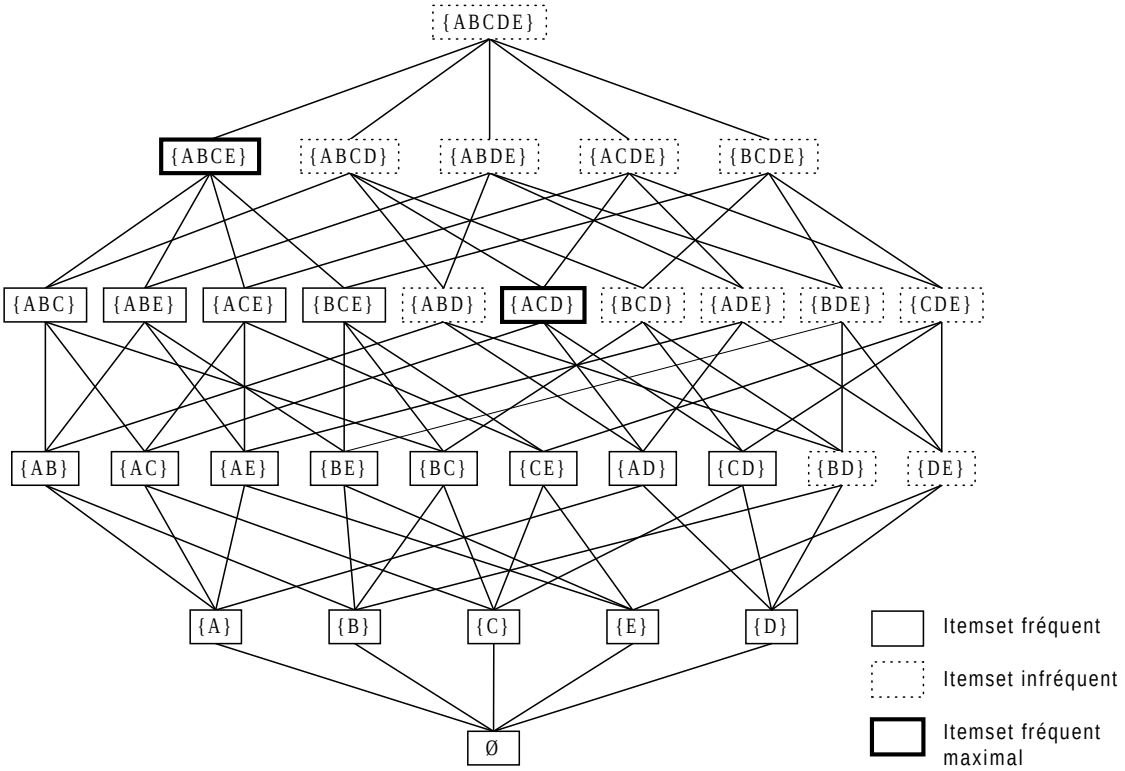


FIG. 2.10 – Treillis des itemsets fréquents maximaux dans le contexte $\mathcal{D}$ pour *minsupport* = 1/6.

Selon les propriétés 2.1 et 2.2, tous les sous-ensembles d'un itemset fréquent sont fréquents et tous les sur-ensembles d'un itemset infréquent sont infréquents. L'ensemble des itemsets fréquents maximaux forme donc une bordure au-dessous de laquelle tous les itemsets sont fréquents et au-dessus de laquelle tous les itemsets sont infréquents. Cette bordure correspond à la *bordure positive* de l'ensemble des itemsets fréquents définie par Toivonen [Toi96b]. Utilisant la propriété de bordure positive, le problème de l'extraction des itemsets fréquents peut se décomposer en deux sous-problèmes :

1. Extraire les itemsets fréquents maximaux dans $\mathcal{B}$, c'est à dire les itemsets dont le support est supérieur ou égal à *minsupport* et dont tous les sur-ensembles sont infréquents.

2. Déterminer les supports de tous les sous-ensembles des itemsets fréquents maximaux en réalisant un balayage de $\mathcal{B}$. Selon les propriétés 2.1 et 2.2, ces itemsets constituent l'ensemble des itemsets fréquents dans $\mathcal{B}$ pour *minsupport*.

Le second sous-problème consiste à réaliser un balayage du contexte en incrémentant le support de tous les itemsets contenus dans chaque objet parcouru. L'extraction des itemsets fréquents maximaux est donc le sous-problème prédominant pour l'efficacité des algorithmes basés sur cette décomposition.

Plusieurs algorithmes d'extraction des itemsets fréquents maximaux ont été proposés dans la littérature. Ce sont des algorithmes itératifs qui réalisent simultanément un parcours du bas vers le haut et un parcours du haut vers le bas du treillis des itemsets. Le parcours du bas vers le haut est une recherche par niveaux identique à celle utilisée par les algorithmes d'extraction des itemsets fréquents. Le parcours du haut vers le bas a pour but d'identifier rapidement des itemsets fréquents maximaux de grande taille. Lors d'une même itération, pendant laquelle est réalisé un balayage du contexte, l'algorithme « avance » d'un niveau du bas vers le haut et de un ou plusieurs niveaux du haut vers le bas. Lorsque un itemset fréquent maximal est identifié, tous ses sous-ensembles n'ont plus à être considérés lors de la recherche du bas vers le haut car selon la propriété 2.1 ce sont des itemsets fréquents. Le principe de cette approche est schématisé dans la figure 2.11. Dans le cas de contextes pour lesquels les plus grands itemsets fréquents sont très larges, c'est à dire pour une valeur de μ élevée, cette approche permet de limiter le nombre d'itérations réalisées par rapport aux algorithmes d'extraction des itemsets fréquents. Les algorithmes d'ex-

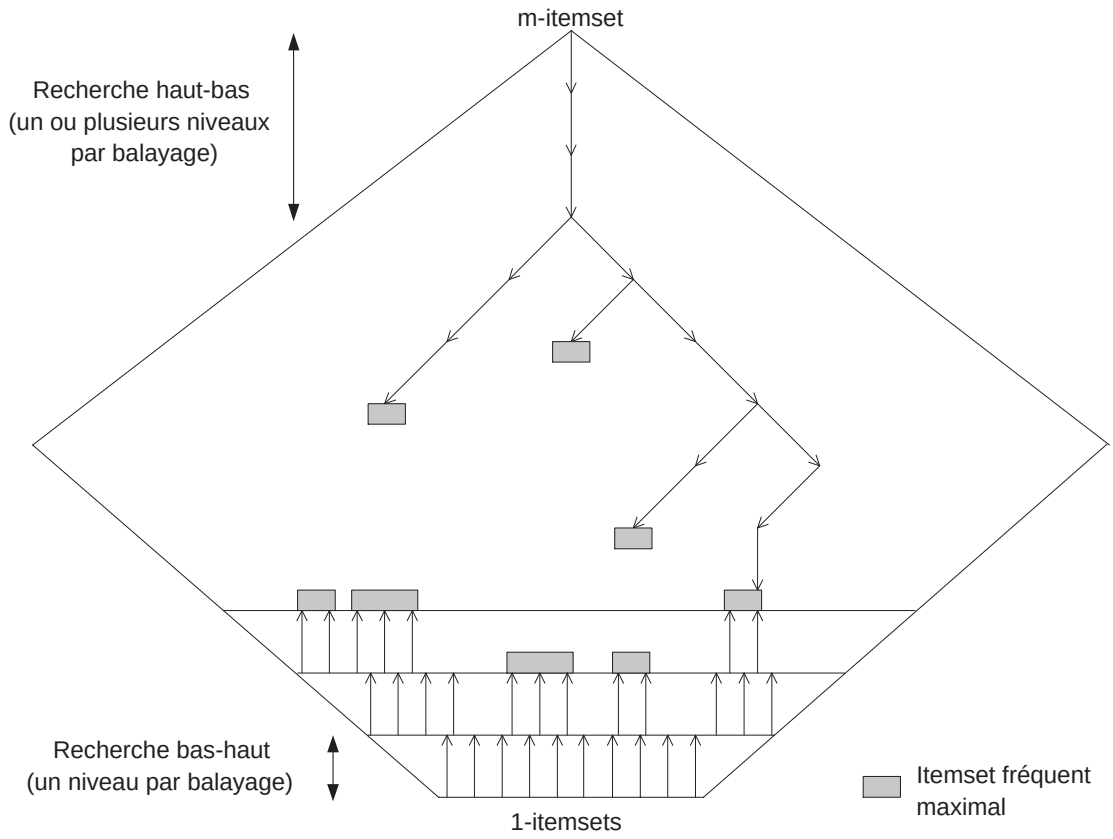


FIG. 2.11 – Principe des algorithmes d'extraction des itemsets fréquents maximaux.

traction des itemsets fréquents maximaux Pincer-Search, MaxEclat et MaxClique sont présentés brièvement dans les sections 2.3.1 et 2.3.2. L'algorithme Max-Miner [Bay98], dont les résultats expérimentaux montrent qu'il est le plus efficace en temps d'exécution, est présenté en détail dans la section 2.3.3.

2.3.1 Pincer-Search

Lin et Kedem [LK98, Lin98] ont proposés un algorithme d'extraction des itemsets fréquents maximaux, nommé Pincer-Search, qui est une extension de l'algorithme Apriori. Dans Pincer-Search, la procédure Apriori-Gen de génération des itemsets candidats est étendue afin de générer un ensemble d'itemsets maximaux candidats utilisés pour le parcours du haut vers le bas. Cet ensemble est initialisé après le premier balayage avec un candidat unique contenant tous les items fréquents dans

le contexte. Durant chaque itération, les supports des itemsets candidats et des itemsets maximaux candidats sont calculés lors du balayage du contexte. Après chaque itération, chaque itemset maximal candidat qui contient un itemset candidat infréquent est remplacé par ses sous-ensembles ne contenant pas l'itemset candidat infréquent. Ces nouveaux itemsets maximaux candidats sont créés en supprimant un item de l'itemset maximal candidat d'origine et en gardant l'itemset résultant si il n'est pas un sous-ensemble d'un autre itemset maximal candidat. Ce processus est répété jusqu'à ce que aucun itemset maximal candidat ne contienne un itemset candidat infréquent. Lorsqu'un itemset maximal candidat est fréquent, tous ses sous-ensembles sont supprimés de l'ensemble des itemsets candidats. Les itérations cessent lorsque plus aucun itemset candidat ne peut être généré et tous les itemsets fréquents maximaux sont alors connus.

2.3.2 MaxEclat et MaxClique

Zaki et al. ont proposés deux algorithmes d'extraction des itemsets fréquents maximaux nommés MaxEclat et MaxClique [ZPOL97, Zak98]. Ces algorithmes utilisent une représentation du contexte d'extraction par listes de OID : à un item i est associée la liste des OID des objets du contexte contenant i. La représentation du contexte $\mathcal{D}$ par listes de OID est présentée dans la table 2.6. Cette représenta-

Item	OID				
A	1	3	5		
B	2	3	4	5	6
C	1	2	3	5	6
D	1				
E	2	3	4	5	6

TAB. 2.6 – Représentation du contexte $\mathcal{D}$ par listes de OID.

tion du contexte d'extraction est construite et stockée sur le disque après le premier balayage. Les items infréquents, pour lesquels la liste de OID associée est de taille inférieure au seuil minimal de support *minsupport*, peuvent ainsi être supprimés du contexte pour les itérations ultérieures. Les calculs des supports des itemsets se font ensuite par intersections des listes de OID des items inclus dans l'itemset.

L'algorithme MaxEclat est basé sur les *classes d'équivalence* des itemsets. Deux k-itemsets font partie de la même classe d'équivalence s'ils partagent les mêmes $k-1$ premiers items. À partir de la seconde itération de la recherche du bas vers le haut, les itemsets candidats déterminés fréquents durant l'itération sont utilisés afin de créer un ensemble d'itemsets maximaux candidats. Pour une itération k, tous les k-itemsets fréquents de F_k possédant les mêmes $k-1$ premiers items (appartenant à la même classe d'équivalence) sont combinés afin de créer un itemset maximal candidat. Si par exemple à la fin de la seconde itération l'ensemble de 2-itemsets fréquents est {{AB}, {AC}, {AE}, {BC}, {BE}, {BF}, {CE}}, l'ensemble des itemsets maximaux candidats sera {{ABCE}, {BCEF}}. L'ensemble ainsi généré contient chaque itemset maximal fréquent de taille supérieure à k, ou bien un de ses sur-ensembles [ZPOL97]. Les supports des itemsets maximaux candidats sont calculés lors de l'itération suivante.

L'algorithme MaxClique utilise la même démarche que l'algorithme MaxEclat mais est basé sur les *cliques maximales* des *hypergraphes uniformes*. À tout ensemble F_k de k-itemsets fréquents peut être associé un hypergraphe uniforme dont les sommets sont les items des k-itemsets et dont les arcs relient les items contenu dans le même k-itemset. Une clique maximale d'un hypergraphe est un ensemble maximal (au sens de l'inclusion) de sommets de l'hypergraphe tous reliés entre eux. Il a été démontré que les itemsets fréquents maximaux de taille supérieure à k sont des sous-ensembles des cliques maximales des hypergraphes uniformes associé à l'ensemble des k-itemsets fréquents [ZPOL97]. Lors d'une itération k consécutive à la seconde itération, les k-itemsets fréquents de F_k appartenant à la même clique de l'hypergraphe associé à F_k sont combinés afin de générer un itemset maximal candidat. Cela signifie qu'un itemset maximal candidat est créé si tous ses sous-ensembles de taille k sont des k-itemsets fréquents. Pour l'ensemble de 2-itemsets fréquents {{AB}, {AC}, {AE}, {BC}, {BE}, {BF}, {CE}}, l'ensemble des itemsets maximaux candidats généré sera {{ABCE}}. Le candidat {BCEF} ne sera pas créé car ses sous-ensembles {CF} et {EF} ne sont pas fréquents. Le nombre d'itemsets maximaux candidats générés par MaxClique est inférieur au nombre de ceux générés par MaxEclat. Toutefois, cette plus grande « précision » des itemsets maximaux candidats entraîne un coût supplémentaire dû au nombre d'opérations requises pour les générer.

Nicolas Pasquier, LIMOS

2.3.3 Max-Miner

L'algorithme Max-Miner a été proposé par Bayardo [Bay98] en 1998. Il effectue simultanément une recherche du bas vers le haut et une recherche du haut vers le bas dans le treillis des itemsets fréquents. La recherche du bas vers le haut est une recherche par niveaux dont le résultat pour chaque itération k (liste des k-itemsets fréquents) est utilisé pour la recherche du haut vers le bas. La recherche du haut vers le bas est réalisée en associant à chaque itemset candidat la liste des items qui ajoutés au candidat peuvent donner un itemset fréquent. Étant donné un ordre défini sur les items, cette liste contient tous les items fréquents dans le contexte qui sont plus grands dans l'ordre que le plus grand item contenu dans le candidat. Cette liste est appelée *liste d'items extensions* du candidats. L'union de chaque itemset candidat avec sa liste d'item extensions constitue l'ensemble d'itemsets maximaux candidats. Après chaque itération, les items de cette liste dont l'union avec l'itemset candidat donne un itemset infréquent sont supprimés de la liste. Au début de l'algorithme, l'ordre sur les items considéré est l'ordre lexicographique. Après le premier balayage du contexte, les items sont ordonnés par support croissant afin de diminuer le coût des opérations sur les candidats.

Le pseudo-code de l'algorithme est présenté dans l'algorithme 2.11. Les notations utilisées sont présentées dans la table 2.7. Chaque élément c de l'ensemble C est un

C	Ensemble de groupes candidats. Chaque élément c de cet ensemble possède - un champ itemset candidat $h(c)$ et un champ $support(h(c))$; - un champ liste d'items extensions $t(c)$; - un champ $support(h(c) \cup t(c))$; - pour chaque item $i \in t(c)$ un champ $support(h(c) \cup \{i\})$.
FM	Ensemble des itemsets fréquents maximaux parmi les itemsets fréquents découverts.

TAB. 2.7 – Notations utilisées dans l'algorithme Max-Miner.

groupe candidat contenant deux itemsets. Le premier dénoté $h(c)$ est un itemset candidat, c'est à dire un itemset dont tous les sous-ensembles sont fréquents et qui est lui-même potentiellement fréquent. Le second dénoté $t(c)$ est la liste des items extensions de $h(c)$. Cette liste contient les items fréquents du contexte dont l'union avec $h(c)$ peut donner un itemset fréquent. Prenons par exemple un ensemble

d'items $\mathcal{I}$ = {A,B,C,D,E,F}. Pour un candidat c = {AB} et si les items C, E et F sont fréquents et l'item D est infréquent dans le contexte, la liste d'items extensions de c sera {C,E,F}.

Lors du balayage du contexte réalisé pour calculer le support du groupe candidat $c \in C$, ce sont les supports des itemsets $h(c)$, $h(c) \cup t(c)$ et $h(c) \cup \{i\}$ pour chaque $i \in t(c)$ qui sont calculés. Les supports des itemsets autres que $h(c)$ sont utilisés pour élaguer l'ensemble de groupes candidats. Si $h(c) \cup t(c)$ est un itemset fréquent alors tous ses sous-ensembles sont fréquents mais non maximaux et donc $h(c) \cup t(c)$ est un itemset fréquent maximal et c peut être supprimé des groupes candidats. Si par contre un itemset $h(c) \cup \{i\}$ pour $i \in t(c)$ est infréquent, alors tous les sur-ensembles de $h(c) \cup \{i\}$ sont infréquents et i peut être supprimé de $t(c)$ avant de créer les candidats de l'itération suivante. Continuant l'exemple précédent, lors du calcul du support du candidat c = {AB}, les supports des itemsets {ABCEF}, {ABC}, {ABE} et {ABF} seront également calculés. Si {ABCEF} est fréquent, il s'agit d'un itemset fréquent maximal et tous ses sous-ensembles n'ont plus besoin d'être considérés. Sinon, si par exemple {ABC} et {ABE} sont fréquents et {ABF} est infréquent, deux nouveaux groupes candidats sont crées à partir de {ABC} et {ABE}.

Durant la première phase de l'algorithme (lignes 1 et 2), l'ensemble C de groupes candidats est initialisé avec l'ensemble vide et l'ensemble FM d'itemsets fréquents maximaux est initialisé avec le 1-itemset possédant le plus grand support dans le contexte. L'initialisation de FM est réalisée par la procédure Gen-Initial-Groups. Durant chacune des itérations suivantes (lignes 3 à 16), un balayage du contexte est réalisé (ligne 4), les supports des groupes candidats sont calculés (ligne 5), les itemsets fréquents maximaux découverts sont insérés dans FM (lignes 6 à 8) et les candidats de l'itération suivante sont créés (lignes 9 à 15) comme suit. Un ensemble C_{new} est créé et initialisé avec l'ensemble vide (ligne 9) et pour chaque groupe candidat $c \in C$, les groupes candidats de l'itération suivante dérivés de c sont insérés dans C_{new} en utilisant la procédure Gen-Sub-Nodes (ligne 11). Cette procédure est également utilisée pour insérer dans FM l'itemset fréquent qui est maximal vis-à-vis du groupe candidat c. Cet itemset peut ne pas être un itemset fréquent maximal final. Lorsque tous les groupes candidats ont été considérés, l'ensemble C_{new} devient le nouvel ensemble candidat pour l'itération suivante (ligne 13), les itemsets qui ne sont pas maximaux dans FM sont supprimés de FM (ligne 14) et les groupes

ALG. 2.11 Extraction des itemsets fréquents maximaux avec Max-Miner.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ;
Sortie : ensemble FM des itemsets fréquents maximaux ;

1) $C \leftarrow \varnothing$;
2) $FM \leftarrow$ Gen-Initial-Groups$(\mathcal{B}, C, minsupport)$;
3) **tant que** $C \neq \varnothing$ **faire**
4) **lire** contexte $\mathcal{B}$;
5) Support-Count$(\mathcal{B}, C)$;
6) **pour chaque** candidat $c \in C$ **tel que** $h(c) \cup t(c)$ est fréquent **faire**
7) $FM \leftarrow FM \cup \{h(c) \cup t(c)\}$;
8) **fin pour**
9) $C_{new} \leftarrow \varnothing$;
10) **pour chaque** candidat $c \in C$ **tel que** $h(c) \cup t(c)$ est infréquent **faire**
11) $FM \leftarrow FM \cup$ Gen-Sub-Nodes$(c, C_{new}, minsupport)$;
12) **fin pour**
13) $C \leftarrow C_{new}$;
14) **supprimer** de FM les itemsets f **tel que** $\exists f' \in FM$ avec $f \subset f'$;
15) **supprimer** de C les groupes c **tel que** $\exists f' \in FM$ avec $h(c) \cup t(c) \subset f'$;
16) **fin tant que**
17) **retourner** FM ;

candidats de C qui ont un sur-ensemble dans FM sont supprimés de C (ligne 15). Les itérations de l'algorithme cessent lorsque aucun groupe candidat ne peut être créé et l'algorithme retourne l'ensemble FM qui contient tous les itemsets fréquents maximaux du contexte (ligne 17).

Procédure Gen-Inital-Groups($\mathcal{B}, C$, *minsupport*) La procédure Gen-Inital-Groups reçoit un contexte $\mathcal{B}$, un ensemble C de groupes candidats et un seuil minimal de support *minsupport* comme paramètres. Elle initialise l'ensemble C des itemsets candidats avec les 1-itemsets fréquents et leur liste d'items extensions et retourne le 1-itemset possédant le plus grand support. Cet itemset sera utilisé pour initialiser l'ensemble FM. Le pseudo-code de la procédure est présenté dans l'algorithme 2.12.

Afin d'améliorer l'efficacité de l'élagage des groupes candidats c pour lesquels

ALG. 2.12 Initialisation des groupes candidats avec Gen-Inital-Groups.

Entrée : contexte $\mathcal{B}$; ensemble C de groupe de candidats ; seuil minimal de support *minsupport* ;

Sortie : ensemble C initialisé ; 1-itemset possédant le plus grand support ;

1) $FM_1 \leftarrow \{$1-itemsets fréquents dans $\mathcal{B}$ avec liste d'items extensions$\}$;
2) **ordonner** les items contenus dans FM_1 par supports croissants ;
3) **pour chaque** item $i \in FM_1$ autre que le plus grand item dans l'ordre **faire**
4) **créer** un groupe candidat c ;
5) $h(c) \leftarrow \{i\}$;
6) $t(c) \leftarrow \{i' \in \mathcal{I} \mid i < i'$ dans l'ordre sur les items$\}$;
7) $C \leftarrow C \cup \{c\}$;
8) **fin pour**
9) **Retourner** l'itemset $f \in FM_1$ contenant le plus grand item dans l'ordre ;

$h(c) \cup t(c)$ est fréquent, il est préférable de maximiser le nombre de groupes candidats possédant cette propriété. Dans cette optique, Max-Miner utilise une stratégie d'ordonnancement des items identique à celle proposée par Brin et al. dans DIC 2.2.5. L'objectif est de faire apparaître les items les plus fréquents dans le contexte dans le plus grand nombre de groupes candidats car les items les plus fréquents sont les plus susceptibles de faire partie des itemsets fréquents maximaux. En ordonnant les items par ordre croissant de leurs supports, les items les plus fréquents seront les derniers dans l'ordre et apparaîtront donc dans le plus grand nombre de groupes candidats.

La procédure commence par créer un ensemble FM_1 contenant tous les 1-itemsets $\{i\}$ fréquents dans $\mathcal{B}$ avec pour chacun la liste des items fréquents suivant i dans l'ordre lexicographique (ligne 1). Les items contenus dans FM_1 sont ensuite ordonnés par ordre croissant de leurs supports (ligne 2) et tous ces items, à l'exception du plus grand dans l'ordre défini, sont considérés successivement en créant un groupe candidat pour chacun de ces items (lignes 3 à 8). La procédure retourne le 1-itemset fréquent de FM_1 contenant le plus grand item dans l'ordre (ligne 9).

Procédure Gen-Sub-Nodes(c, C, *minsupport*) La procédure Gen-Sub-Nodes reçoit un groupe candidat c, un ensemble C de groupes candidats et un seuil minimal

de support *minsupport* comme paramètres. Elle met à jour l'ensemble C de groupes candidats avec les groupes générés à partir du groupe candidat c. Elle retourne également l'itemset fréquent sur-ensemble de c de taille $|c|+1$ possédant le plus grand support. Si un tel itemset n'existe pas, elle retourne $h(c)$. Le pseudo-code de la procédure est présenté dans l'algorithme 2.13.

ALG. 2.13 Génération des groupes candidats avec Gen-Sub-Nodes.

Entrée : groupe candidat c ; ensemble C de groupes candidats ; seuil minimal de support *minsupport* ;

Sortie : ensemble C de groupes candidats mis à jour ; itemset fréquent $h(c) \cup \{i\}$ possédant le plus grand support pour $i \in t(c)$ ou bien $h(c)$;

1) **pour chaque** item $i \in t(c)$ **faire**
2) **si** $h(c) \cup \{i\}$ est infréquent **alors** $t(c) \leftarrow t(c) \setminus \{i\}$;
3) **fin pour**
4) **ordonner** les items i de $t(c)$ par supports de $h(c) \cup \{i\}$ croissants ;
5) **pour chaque** item $i \in t(c)$ autre que le plus grand itemset dans $t(c)$ **faire**
6) **créer** un groupe candidat c' ;
7) $h(c') \leftarrow h(c) \cup \{i\}$;
8) $t(c') \leftarrow \{i' \in t(c) \mid i < i' \text{ dans } t(c)\}$;
9) $C \leftarrow C \cup \{c'\}$;
10) **fin pour**
11) **si** $(t(c) = \varnothing)$ **alors retourner** $h(c)$;
12) **sinon retourner** $h(c) \cup \{i\}$ avec i plus grand item dans $t(c)$;

La première partie de la procédure (lignes 1 à 3) supprime de la liste d'items extension de c les items i pour lesquels $h(c) \cup \{i\}$ est infréquent. Les items i restant dans $t(c)$ sont ensuite ordonnés par ordre croissant des supports de $h(c) \cup \{i\}$ (ligne 4). Pour chacun de ces items i, à l'exception du plus grand dans l'ordre défini, un nouveau groupe candidat c' est créé dans C avec pour itemset candidat $h(c') = h(c) \cup \{i\}$ (lignes 5 à 10). Si aucun des itemsets $h(c) \cup \{i\}$ pour $i \in t(c)$ n'est fréquent ($t(c)$ est vide) alors la procédure retourne $h(c)$ (ligne 11). Sinon, elle retourne l'itemset $h(c) \cup \{i\}$ dont le support est le plus élevé (ligne 12) et donc dont la liste d'items extensions est vide car i est le plus grand item dans $t(c)$.

Structures de données L'algorithme Max-Miner utilise un arbre de hachage identique à celui utilisé par l'algorithme Apriori pour stocker les groupes candidats. Un noeud feuille de l'arbre contient les itemsets candidats $h(c)$ des groupes candidats $c \in C$ pour lesquels le noeud feuille a été atteint en appliquant successivement la fonction de hachage à chacun des items $i \in c$. À chaque itemset candidat c dans un noeud feuille sont associés la liste $t(c)$ d'items extensions de $h(c)$ et $|t(c)| + 2$ champs supports. Les itemsets candidats contenus dans un objet o du contexte sont déterminés en appliquant récursivement la fonction de hachage a chacun des items $i \in o$. Pour chacun de ces itemsets candidats c, les items i de la liste d'items extension $t(c)$ sont parcourus un à un et le support de $h(c) \cup i$ est incrémenté si i est contenu dans l'objet o. Si tous les items de $t(c)$ sont contenus dans l'objet, le support de $h(c) \cup t(c)$ est également incrémenté. Les arbres de hachage sont également utilisés afin d'identifier efficacement les groupes candidats pour lesquels $h(c) \cup t(c)$ possède un sur-ensemble dans FM (ligne 14).

Exemple 2.6 L'application de l'algorithme Max-Miner au contexte $\mathcal{D}$ pour un seuil minimal de support *minsupport* de 2/6 est représentée dans la figure 2.12. L'algorithme réalise deux balayages du contexte afin d'extraire l'itemset fréquent maximal {ABCE} et un balayage supplémentaire pour déterminer le support de tous les sous-ensembles de {ABCE}. En re-ordonnant les items par ordre croissant des supports des 1-itemsets correspondants dans FM_1, l'ordre obtenu est identique à l'ordre lexicographique puisque nous avons *support*(A) $<$ *support*(B) $=$ *support*(C) $=$ *support*(E).

Calcul de bornes inférieures des supports Le calcul de bornes inférieures des supports des itemsets permet de limiter le nombre de groupes candidats générés par la procédure Gen-Sub-Nodes. Lors de la création des groupes candidats de l'itération suivante à partir d'un groupe candidat c, les items $i \in t(c)$ de la liste d'items extension sont parcourus un à un dans l'ordre défini et un itemset candidat $h(c) \cup \{i\}$ est généré. Pour un itemset candidat c_2 généré après un itemset candidat c_1, nous aurons la propriété suivante : $h(c_2) \cup t(c_2) \subset h(c_1) \cup t(c_1)$. Donc, si $h(c_1) \cup t(c_1)$ est un itemset fréquent, $h(c_2) \cup t(c_2)$ ne peut être un itemset fréquent maximal et il est inutile de considérer le groupe candidat c_2. Le calcul de la borne inférieure du support de $h(c_1) \cup t(c_1)$ est réalisé lors de la génération du groupe candidat c_1 à

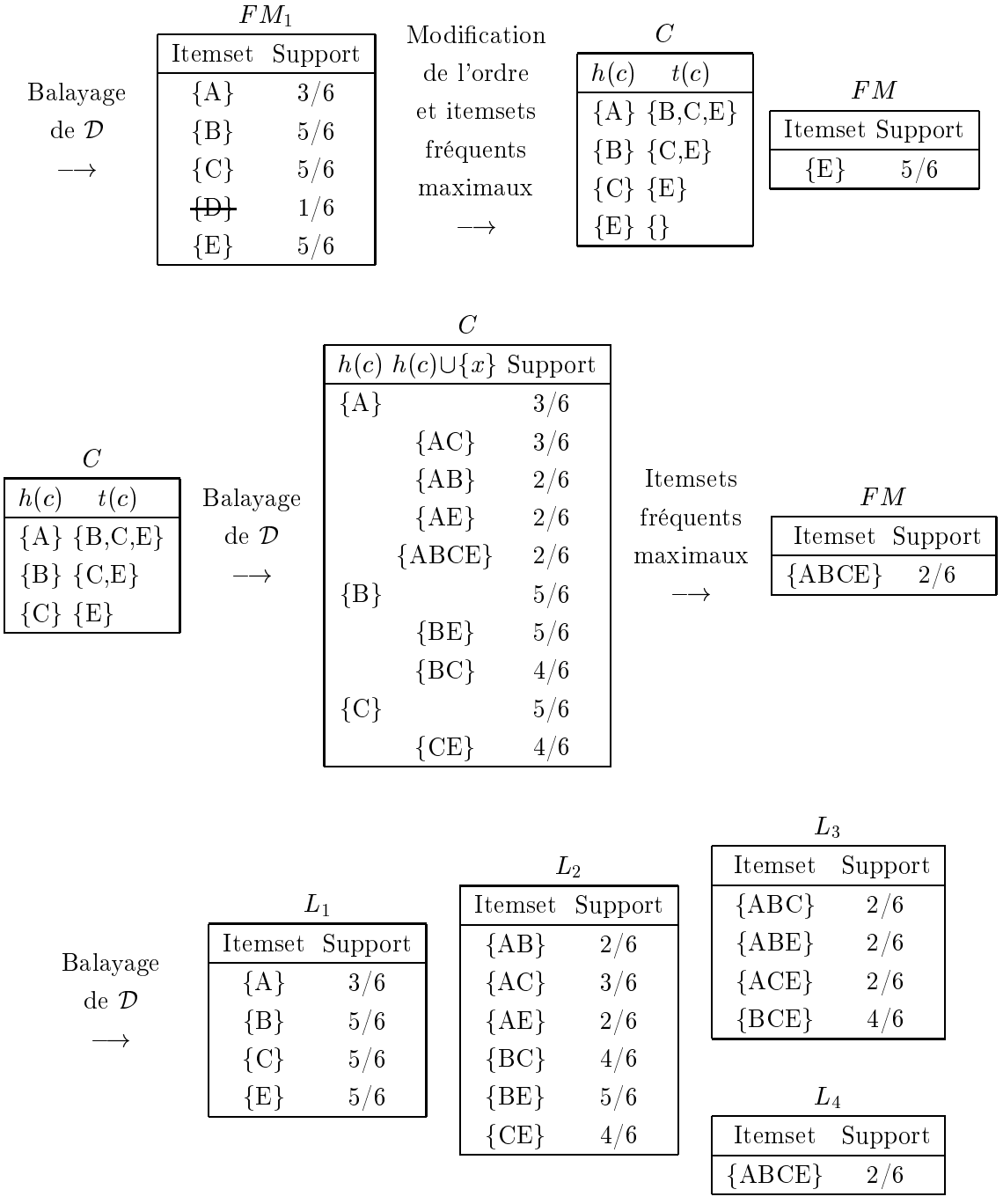


FIG. 2.12 – Extraction des itemsets fréquents dans le contexte $\mathcal{D}$ avec Max-Miner pour $minsupport = 2/6$.

partir de c. Si celle-ci est supérieure ou égale au seuil minimal de support *minsupport*, il n'est pas nécessaire de générer les groupes candidats suivants c_1 et la procédure Gen-Sub-Nodes s'interrompt et retourne $h(c_1) \cup t(c_1)$.

Soit $drop(l, x)$ le nombre d'objets du contexte qui sont « éliminés » du groupe d'objet contenant l'itemset l lorsque on lui ajoute l'item $i \notin l$. Nous avons $drop(l, i) = support(l) - support(l \cup \{i\})$ et donc $support(l \cup \{i\}) = support(l) - drop(l, i)$. Le théorème 2.1 est la généralisation de cette propriété afin de calculer une borne inférieure du support de l'itemset $h(c_1) \cup t(c_1)$ en utilisant les supports de ses sous-ensembles $h(c) \cup \{i\}$ déjà calculés lors de l'itération précédente.

Théorème 2.1 (Borne inférieure du support)
Soit c_1 un groupe candidat généré à partir du groupe candidat $c \in C$. Nous avons donc $h(c_1) = h(c) \cup \{i\}$ avec $i \in t(c)$ et $t(c_1) = \{\bigcup i' \in t(c) \mid i' >_{t(c)} i \}$. L'équation suivante fournit une borne inférieure du support de l'itemset $h(c_1) \cup t(c_1)$.

$$support(h(c_1)) - \sum_{i' \in t(c_1)} drop(h(c), i').$$

2.3.4 Discussion

Les algorithmes présentées dans cette section recherchent les itemsets fréquents maximaux simultanément du bas vers le haut et du haut vers le bas parmi les itemsets fréquents en maintenant un ensemble contenant les plus grands itemsets fréquents possibles. Les algorithmes d'extraction des itemsets fréquents par niveaux requièrent μ itérations afin de déterminer tous les itemsets fréquents, μ étant la taille des itemsets fréquents (maximaux) les plus longs. En identifiant ces itemsets par la recherche du haut vers le bas, les algorithmes d'extraction des itemsets fréquents maximaux réduisent le nombre total d'itérations nécessaires à l'extraction des itemsets fréquents dans le cas ou μ est élevé. De plus, lorsqu'un itemset fréquent maximal est identifié, ses sous-ensembles n'ont plus à être considérés ce qui réduit le nombre d'itemsets candidats générés et donc les temps de calcul CPU. Les résultats des expérimentations confirment la réduction des temps d'extraction des itemsets fréquents lorsque un algorithme d'extraction des itemsets fréquents maximaux est utilisé. Ils montrent également que l'algorithme Max-Miner est globalement plus performant que les algorithmes Pincer-Search, MaxEclat et MaxClique en termes de temps d'exécution.

La méthode utilisée dans l'algorithme Pincer-Search pose deux problèmes importants. Le premier est la détermination, à la fin de chaque itération, des itemsets maximaux candidats contenant des itemsets infréquents qui est un problème NP-difficile [Bay98]. Le second est la génération des itemsets candidats de l'itération suivante qui requiert de nombreux tests d'inclusions dans les itemsets fréquents maximaux du fait de la suppression des itemsets candidats qui sont des sous-ensembles de ces itemsets fréquents maximaux. Ces deux problèmes, qui nécessitent des temps de calcul CPU importants, entraînent dans certains cas des temps de réponse nettement supérieurs pour l'algorithme Pincer-Search par rapport à l'algorithme Max-Miner.

L'algorithme MaxEclat est soumis au problème du manque de précision des itemsets maximaux candidats qu'il utilise. Ces candidats, qui sont générés en combinant les itemsets fréquents identiques sauf pour leur dernier item, sont des sur-ensembles des itemsets fréquents maximaux dont de nombreux items doivent être supprimés pour obtenir les itemsets fréquents maximaux. En conséquence, l'algorithme MaxEclat doit réaliser davantage d'itérations que les algorithmes Max-Miner et Pincer-Search pour identifier les itemsets fréquents maximaux. Chaque itération correspondant à un balayage du contexte et au calcul du support de tous les candidats, ces itérations représentent une différence non négligeable dans les temps d'exécution.

L'algorithme MaxClique est soumis à un problème de performance dû aux temps de calcul requis pour la génération des itemsets maximaux candidats qu'il utilise. Il génère les cliques maximales du graphe dont les sommets sont les items fréquents et les arcs les k-itemsets fréquents. Les itemsets résultant, qui sont des sur-ensembles des itemsets fréquents maximaux, sont les itemsets maximaux candidats utilisés pour la recherche du haut vers le bas. Le problème de l'énumération des cliques maximales d'un graphe étant un problème NP-difficile, le nombre d'opérations nécessaires à leur génération est très important et les temps de réponses de l'algorithme se dégradent considérablement lorsque les itemsets fréquents (et donc les cliques) sont longs.

L'algorithme Max-Miner a été proposé concurremment aux algorithmes Close et A-Close et, ne disposant pas d'implémentation complète de cet algorithme, nous n'avons pas pu réaliser d'expérimentations permettant de comparer les performances de ces trois algorithmes. Les résultats des expérimentations de comparaison de l'algorithme Max-Miner avec les algorithmes d'extraction des itemsets fréquents montrent qu'il permet de réduire les temps d'extraction des itemsets fréquents dans le cas de valeurs de μ élevées, c'est à dire si les plus longs itemsets fréquents maximaux

contiennent un nombre important d'items. Les expérimentations des algorithmes Close et A-Close montrent également que ces deux algorithmes réduisent les temps d'extraction des itemsets fréquents pour de tels jeux de données. Toutefois, l'algorithme Max-Miner nécessite une quantité importante de mémoire afin de stocker les informations concernant les groupes candidats de chaque itération. De plus, cet algorithme est soumis au problème de l'extraction de règles d'association à partir de données denses et fortement corrélées lié à la taille moyenne des itemsets fréquents qui est élevée pour les jeux de données de ce type.

Chapitre 3

Génération et réduction des règles d'association

Sommaire

3.1 Introduction

La génération des règles d'association est réalisée à partir de l'ensemble F des itemsets fréquents dans le contexte d'extraction pour le seuil minimal de support *minsupport*. Une règle d'association r est une relation entre itemsets de la forme $r : l_2 \rightarrow (l_1 - l_2)$[1] dans laquelle l_1 et l_2 sont des itemsets fréquents tels que $l_2 \subset l_1$. Les itemsets l_2 et $(l_1 - l_2)$ sont appelés respectivement l'antécédent et la conséquence de la règle r. Les règles d'association valides sont celles dont la confiance, résultat du rapport $support(l_1)/support(l_2)$, est supérieure ou égale au seuil minimal de confiance *minconfiance*.

Le problème de la génération des règles d'association est un problème exponentiel dans la taille des itemsets fréquents. En effet, à partir d'un itemset fréquent f de taille supérieure à un, $2^{|f|} - 2$ règles d'association non triviales peuvent être générées. Toutefois, cette génération est réalisée de manière directe, sans accéder au jeu de données, et les temps d'exécution de cette étape sont faibles comparés au temps d'exécution de l'extraction des itemsets fréquents. Un algorithme efficace de génération des règles d'association à été proposé par Agrawal et al. dans [AS94]. Cet algorithme, qui donne des temps de réponse acceptables dans la majorité des cas, est présenté dans la section 3.2.

Pour un ensemble F d'itemsets fréquents extraits, le nombre de règles d'association valides pouvant être générées à partir de F est $\sum_{f \in F} 2^{|f|} - 2$. Le nombre d'itemsets fréquents extraits et leur taille moyenne étant élevés dans la plupart des jeux de données, le nombre de règles d'association générées varie en général de plusieurs dizaines de milliers à plusieurs millions. Ce nombre important de règles d'association extraites constitue un problème majeur pour la pertinence et l'utilité du résultat, problème qui est accentué par la présence de nombreuse règles redondantes du point de l'information convoyée. Considérons par exemple l'ensemble des règles d'association extraites du contexte $\mathcal{D}$ pour *minsupport* = 2/6 et *minconfiance* = 1/2 qui contient les règles A → BCE, A → BC, A → BE, A → CE, A → B et A → E. Toutes ces règles possèdent le même antécédent ainsi que des supports (2/6)

[1] Afin de faciliter la lecture des exemples, une règle d'association {AB} → {CD} est notée AB → CD.

et des confiances (2/3) identiques. Clairement, les cinq dernières règles sont redondantes vis à vis de la règle A → BCE du point de vue de l'information convoyée : elles ne fournissent aucune information supplémentaire par rapport à cette règle.

La réduction de l'ensemble de règles d'association générées afin d'améliorer la pertinence du résultat à fait l'objet de plusieurs travaux de recherche. Les approches proposées pour résoudre ce problème peuvent se classer en deux catégories : les approches orientées structure de données, présentés dans la section 3.3, et les approches orientées utilisateur, décrites dans la section 3.4. Les diverses aspects de ces deux approches, ainsi que leurs avantages et leurs inconvénients, sont discutés dans les section 3.3.5 et 3.4.4.

3.2 Génération de l'ensemble des règles d'association

Le principe général de la génération de l'ensemble des règles d'association est le suivant. Pour chaque itemset fréquent l_1 dans F de taille supérieure ou égale à deux, tous les sous-ensembles l_2 de l_1 sont déterminés et la valeur du rapport $support(l_1)/support(l_2)$ est calculée. Si cette valeur est supérieure ou égale au seuil de confiance *minconfiance* alors la règle d'association $l_2 \rightarrow (l_1 - l_2)$ est générée. L'algorithme proposé par Agrawal et al. [AS94] se base sur la propriété 3.1 concernant les supports des itemsets afin de réduire le nombre d'opérations réalisées par la génération.

Propriété 3.1 *Étant donné un itemset l, le support d'un sous-ensemble l' de l est supérieur ou égal au support de l.*

Étant donné trois itemsets fréquents l_1, l_2 et l_3 tels que $l_3 \subset l_2 \subset l_1$, nous pouvons déduire de la propriété 3.1 que $support(l_3) \geq support(l_2) \geq support(l_1)$. En conséquence, la confiance de la règle $r' : l_3 \rightarrow (l_1 - l_3)$ est inférieure ou égale à la confiance de la règle $r : l_2 \rightarrow (l_1 - l_2)$. Si la règle r n'est pas valide alors la règle r' ne sera pas valide non plus. Cela signifie que si la règle d'association AC → DE n'est pas valide, alors les règles A → CDE et C → ADE ne sont pas valides non plus et il n'est pas nécessaire de calculer leurs confiances. Cette constatation permet de diminuer le nombre de règles d'association testée par l'algorithme. Réciproquement,

la confiance de la règle $s : (l_1 - l_2) \rightarrow l_2$ est supérieure ou égale à la confiance de la règle $s' : (l_1 - l_3) \rightarrow l_3$. Si la règle s' est valide alors la règle s sera également valide. Cela signifie que si la règle d'association A $\rightarrow$ BC est valide, alors les règles AB $\rightarrow$ C et AC $\rightarrow$ B sont également valides.

F	Ensemble d'itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
H_m	m-itemsets qui sont les conséquences de règles valides générées à partir de l'itemset l_k.

TAB. 3.1 – Notations utilisées dans l'algorithme Gen-Règles.

ALG. 3.1 Génération de l'ensemble des règles d'association avec Gen-Règles.

Entrée : ensemble F d'itemsets fréquents ; seuil minimal de confiance *minconfiance* ;

Sortie : ensemble $\mathcal{AR}$ de règles d'association valides ;

1) **pour chaque** k-itemsets fréquents $l_k \in F$ tel que $k \geq 2$ **faire**
2) $\quad H_1 \leftarrow \{$1-itemsets sous-ensembles de $l_k\}$;
3) $\quad$ **pour chaque** $h_1 \in H_1$ **faire**
4) $\quad\quad$ $confiance(r) \leftarrow support(l_k)/support(l_k - h_1)$;
5) $\quad\quad$ **si** $(confiance(r) \geq minconfiance)$ **alors**
6) $\quad\quad\quad$ $\mathcal{AR} \leftarrow \mathcal{AR} \cup \{r : (l_k - h_1) \rightarrow h_1\}$;
7) $\quad\quad$ **sinon** $H_1 \leftarrow H_1 \setminus \{h_1\}$;
8) $\quad$ **fin pour**
9) $\quad$ **faire** Gen-Rules(l_k, H_1) ;
10) **fin pour**
11) **retourner** $\mathcal{AR}$;

Le pseudo-code de l'algorithme est présenté dans l'algorithme 3.1. Les notations utilisées sont présentées dans la table 3.1. L'algorithme considère successivement chaque itemset fréquent de F de taille supérieure à un (lignes 1 à 10). Pour chacun de ces itemsets l_k, l'ensemble H_1 des itemsets de taille 1 qui sont des sous-ensembles de l_k est généré (ligne 2) et pour chacun de ces itemsets h_1 la règle $(l_k - h_1) \rightarrow h_1$ est générée si sa confiance est supérieure ou égale à *minconfiance* (ligne 4 à 6). Sinon,

si la règle $(l_k - h_1) \rightarrow h_1$ n'est pas valide, alors le 1-itemset h_1 est supprimé de H_1 (ligne 7). Lorsque tous les 1-itemsets de H_1 ont été testés, H_1 contient la liste des 1-itemsets qui sont les conséquences des règles valides générées à partir de l_k. Les règles valides générées à partir de l_k sont les règles dont l'union de l'antécédent et de la conséquence donne l'itemset l_k. La procédure Gen-Rules est alors appelée (lignes 9) afin d'insérer dans $\mathcal{AR}$ les règles valides générées à partir de l_k dont la conséquence contient plus de un item. L'algorithme se termine lorsque tous les k-itemsets fréquents pour $k \geq 2$ ont été considérés. L'ensemble $\mathcal{AR}$ renvoyé par l'algorithme (ligne 11) contient alors toutes les règles d'association valides pour le seuil minimal de confiance *minconfiance* générées à partir de l'ensemble F.

Procédure Gen-Rules(l_k, H_m) La procédure Gen-Rules reçoit un k-itemset fréquent l_k, un ensemble H_m qui contient les m-itemsets qui sont les conséquences de règles valides générées à partir de l_k et un seuil minimal de confiance *minconfiance* comme paramètres. Elle met à jour l'ensemble $\mathcal{AR}$ de règles d'association en y insérant les règles valides générées à partir de l_k dont la conséquence est un $(m+1)$-itemset. Cette procédure est récursive et réalise en fin d'exécution un appel afin de générer à partir de l_k les règles valides dont la conséquence est un $(m+2)$-itemset. Ces appels se répètent récursivement jusqu'à ce que les règles dont la conséquence est un $(|l_k|-1)$-itemset aient été insérées dans $\mathcal{AR}$. Le pseudo-code de la procédure est présenté dans l'algorithme 3.2.

Le premier test de l'algorithme (ligne 1) correspond au test d'arrêt des appels récursifs de la procédure. Ces appels cessent lorsque l'ensemble H_m reçu comme paramètre contient des itemsets de taille $m = |l_k| - 1$. Dans ce cas toutes les règles valides générées à partir de l_k ont été insérées dans $\mathcal{AR}$. Ensuite, l'ensemble H_{m+1} des $(m+1)$-itemsets qui peuvent être des conséquences de règles valides générées à partir de l_k est créé. Cette création est réalisée en appliquant la procédure Apriori-Gen (section 2.2.1) à l'ensemble H_m des m-itemsets qui sont les conséquences de règles valides générées à partir de l_k (ligne 2). Chaque règle dont la conséquence est un $(m+1)$-itemset de H_{m+1} est alors testée (lignes 3 à 8). Si la règle testée est valide, elle est insérée dans $\mathcal{AR}$ (ligne 6). Sinon, le $(m+1)$-itemset qui en est la conséquence est supprimé de H_{m+1} (ligne 7). Cette suppression correspond à la diminution du nombre de règles testées basée sur la propriété 3.1. En effet, si la règle d'association AC $\rightarrow$ DE n'est pas valide, l'itemset DE est supprimé de H_2. Lors de l'appel récursif

ALG. 3.2 Insertion des règles d'association dans $\mathcal{AR}$ avec Gen-Rules.

Entrée : k-itemsets fréquents l_k ; ensemble H_m de m-itemsets conséquences de règles valides générées à partir de l_k ; seuil minimal de confiance *minconfiance* ;

Sortie : ensemble $\mathcal{AR}$ de règles d'association valides augmenté des règles valides générées à partir de l_k dont la conséquence est un $(m+1)$-itemset ;

1) **si** $(k > m+1)$ **alors faire**
2) $H_{m+1} \leftarrow$ Apriori-Gen(H_m) ;
3) **pour chaque** $h_{m+1} \in H_{m+1}$ **faire**
4) $confiance(r) \leftarrow support(l_k)/support(l_k - h_{m+1})$;
5) **si** $(confiance(r) \geq minconfiance)$ **alors**
6) $\mathcal{AR} \leftarrow \mathcal{AR} \cup \{r : (l_k - h_{m+1}) \rightarrow h_{m+1}\}$;
7) **sinon supprimer** h_{m+1} de H_{m+1} ;
8) **fin pour**
9) **faire** Gen-Rules(l_k, H_{m+1}) ;
10) **finsi**

suivant (avec H_2 comme paramètre) les itemsets CDE et ADE ne seront pas créés par Apriori-Gen dans H_3 car DE est un sous-ensemble de CDE et ADE. Les règles A $\rightarrow$ CDE et C $\rightarrow$ ADE ne seront donc pas testées. L'appel récursif à Gen-Rules est réalisé en fin de procédure (ligne 9) avec comme paramètres l'itemset l_k et l'ensemble H_{m+1}.

Exemple 3.1 L'ensemble F des itemsets fréquents dans le contexte $\mathcal{D}$ pour un seuil minimal de support *minsupport* de 3/6 est présenté dans le figure 3.1. La génération de l'ensemble des règles d'association à partir de cet ensemble F pour un seuil minimal de confiance *minconfiance* de 1/2 est présenté dans la figure 3.2.

3.3 Réduction de l'ensemble des règles d'association : approches orientées structure des données

Les approches orientées structure des données se basent sur les propriétés structurelles des règles d'association afin de réduire l'ensemble des règles d'association

F_3

Itemset	Support
{BCE}	4/6

F_2

Itemset	Support
{AC}	3/6
{BC}	4/6
{BE}	5/6
{CE}	4/6

F_1

Itemset	Support
{A}	3/6
{B}	5/6
{C}	5/6
{E}	5/6

FIG. 3.1 – Itemsets fréquents extraits du contexte $\mathcal{D}$ pour $minsupport = 3/6$.

extraites. Nous distinguons quatre catégories différentes parmi ces approches. La première consiste à utiliser une taxonomie, ou hiérarchie de classes, des items afin de générer des règles entre ensembles d'items de différents niveaux dans la taxonomie. Cette approche est présentée dans la section 3.3.1. La seconde est l'utilisation de mesures statistiques autres que la confiance pour déterminer la précision des relations entre itemsets. Les travaux concernant l'utilisation de mesures statistiques autres que la confiance sont présentés dans la section 3.3.2. Dans la troisième, des *mesures de déviation* des règles d'association sont utilisées en plus des mesures de support et confiance. Les mesures de déviation associées aux règles sont calculées en utilisant le support et la confiance de ces règles. Les règles extraites sont celles dont la mesure de déviation est supérieure à un seuil minimal défini par l'utilisateur, qui représentent des variations importantes par rapport à la distribution moyenne ou la distribution attendue des données dans le contexte. Les diverses approches utilisant des mesures de déviation des règles sont décrites dans la section 3.3.3. La quatrième approche consiste à extraire l'ensemble des règles d'association valides et supprimer de cette ensemble les règles redondantes en fonction de la structure syntaxique des règles. Cette approche, qui ne peut être appliquée sans perte d'information que si le contexte possède certaines propriétés, est présentée dans la section 3.3.4.

3.3.1 Règles d'association généralisées

Les *règles d'association généralisées* [SA95], également appelées *règles d'association multi-niveaux* [HF95], sont définies en utilisant une taxonomie des items du contexte d'extraction. Cette taxonomie est un graphe dirigé acyclique dont les sommets sont les items et les arcs sont des relations *is-a* entre deux items. Si il existe un arc d'un item i vers un item i' dans la taxonomie alors i est appelé *père* de i' et

l_k

Itemset	Support
{BCE}	4/6

Génération des règles $\longrightarrow$

1-itemset conséquence

Règle	Confiance
BC → E	4/4
BE → C	4/5
CE → B	4/4

Génération des règles $\longrightarrow$

2-itemset conséquence

Règle	Confiance
B → CE	4/5
C → BE	4/5
E → BC	4/5

l_k

Itemset	Support
{AC}	3/6

Génération des règles $\longrightarrow$

1-itemset conséquence

Règle	Confiance
A → C	3/3
C → A	3/5

l_k

Itemset	Support
{BE}	5/6

Génération des règles $\longrightarrow$

1-itemset conséquence

Règle	Confiance
B → E	5/5
E → B	5/5

l_k

Itemset	Support
{BC}	4/6

Génération des règles $\longrightarrow$

1-itemset conséquence

Règle	Confiance
B → C	4/5
C → B	4/5

l_k

Itemset	Support
{CE}	4/6

Génération des règles $\longrightarrow$

1-itemset conséquence

Règle	Confiance
C → E	4/5
E → C	4/5

FIG. 3.2 – Génération des règles d'association valides dans le contexte $\mathcal{D}$ pour *minsupport* = 3/6 et *minconfiance* = 1/2.

i' est appelé *fils* de i. Si il existe un arc d'un item i vers un item i' dans la fermeture transitive de la taxonomie, i est appelé *ancêtre* de i' et i' est appelé *descendant* de i. Nous notons alors $i \in ancêtre(i')$ et $i' \in descendant(i)$, ce qui signifie que l'item i est une généralisation de l'item i' et que i' est une spécialisation de i. Un item n'est pas un ancêtre de lui-même car le graphe est acyclique.

Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ et une taxonomie $\mathcal{T}$ associée à $\mathcal{B}$. Dans la plupart des cas, les items $i \in \mathcal{I}$ contenus dans les objets du contexte d'extraction sont des sommets feuilles de la taxonomie, c'est à dire tels que $descendant(i) = \varnothing$, et les items qui possèdent des descendants sont des concepts de généralisation des items du contexte.

Exemple 3.2 Un contexte d'extraction $\mathcal{V}$, dérivé d'une base de données de ventes, constitué de six objets et cinq items est représenté dans la table 3.2. Une taxonomie $\mathcal{T}$ définie sur les items du contexte $\mathcal{V}$ est présentée dans la figure 3.3. Les items "Articles", "Sous-vêtements" et "Chaussures" sont des généralisations des items du contexte $\mathcal{V}$. Le contexte $\mathcal{V}$ et la taxonomie $\mathcal{T}$ associée sont utilisés dans la suite comme support pour les exemples d'applications des approches de réduction des règles d'association.

OID	Items		
1	Baskets	Caleçon	
2	T-shirt	Pantalon	
3	Baskets	Pantalon	Caleçon
4	T-shirt	Pantalon	
5	Pantalon	Caleçon	
6	Baskets	T-shirt	

TAB. 3.2 – Contexte d'extraction de règles d'association $\mathcal{V}$.

Les règles d'association généralisées sont des règles d'association entre ensembles d'items pouvant appartenir à différents niveaux de la taxonomie. Elles sont de la forme $r : l_1 \rightarrow l_2$ avec $l_1, l_2 \subset \{\bigcup i \in \mathcal{T}\}$ et $l_1 \cap l_2 = \varnothing$ telles que aucun item de l_2 n'est un ancêtre d'un item de l_1. Celle dernière condition est nécessaire afin que les règles triviales de la forme "$i \rightarrow$ ancêtre(i)" ne soient pas générées. Les règles d'association généralisées valides sont celles dont le support et la confiance sont su-

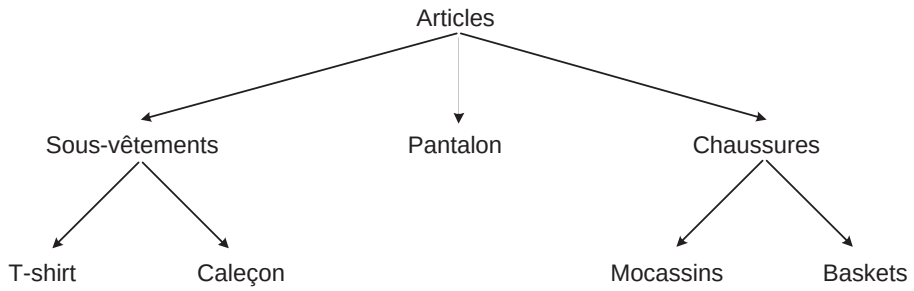


FIG. 3.3 – Taxonomie $\mathcal{T}$ des items du contexte $\mathcal{V}$.

périeurs ou égaux aux seuils minimaux *minsupport* et *minconfiance* respectivement. Le support d'un item n'est pas égal à la somme des supports de ses descendants, car plusieurs de ses descendants peuvent être contenus dans le même objet. Il n'est donc pas possible de dériver les règles d'association généralisées depuis les règles d'association entre ensembles d'items feuilles de la taxonomie.

Les itemsets fréquents, à partir desquels les règles d'association généralisées sont générées, sont constitués des items de la taxonomie $\mathcal{T}$. Ces itemsets sont appelés itemsets généralisés fréquents. L'ensemble $\mathcal{I}' = \{\bigcup i \in \mathcal{T}\}$ des items utilisés pour l'extraction des itemsets généralisés fréquents est donc un sur-ensemble de l'ensemble $\mathcal{I}$ des items contenus dans les objets du contexte. Un item $i \in \mathcal{I}'$ est contenu dans un objet $o \in \mathcal{O}$ si o contient i ou un descendant de i. Le support d'un item $i \in \mathcal{I}'$ est donc inférieur ou égal aux supports de ses ancêtres. Le support d'un itemset $l \subseteq \mathcal{I}'$ est la proportion d'objets du contexte qui contiennent chaque item $i \in l$ ou un de ses descendants. Les itemsets généralisés fréquents sont les itemsets $l \subseteq \mathcal{I}'$ tels que support$(l) \geq$ *minsupport* et aucun item $i \in l$ n'est un ancêtre d'un autre item $i' \in l$.

Exemple 3.3 L'ensemble d'itemsets généralisés fréquents extraits du contexte $\mathcal{V}$ pour un seuil minimal de support de 2/6 est présenté dans la table 3.3. Ces itemsets sont générés à partir de l'ensemble d'items de la taxonomie $\mathcal{I}' =$ {Sous-vêtements, T-shirt, Caleçon, Pantalon, Chaussures, Mocassins, Baskets}. L'item "Articles" n'est pas utilisé car il est ancêtre de tous les autres items. Les règles de la forme "Articles $\rightarrow i$" ne convoient aucune information supplémentaire par rapport aux itemsets $\{i\}$ car leurs confiances sont égales aux supports des itemsets $\{i\}$. L'ensemble de règles d'association généralisées valides dans le contexte $\mathcal{V}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance de 2/3 est présenté dans la table 3.4.

Pour de tels seuils de support et de confiance, 11 règles d'association sont générées à partir des 16 itemsets généralisés fréquents dans le contexte $\mathcal{V}$. Les règles "T-shirt $\rightarrow$ Baskets" et "Caleçon $\rightarrow$ Baskets" ne sont pas générées car leurs supports ne sont pas suffisant contrairement à la règle "Sous-vêtements $\rightarrow$ Baskets".

Itemset généralisé	Support	Itemset généralisé	Support
{T-shirt}	3/6	{Caleçon, Pantalons}	2/6
{Caleçon}	3/6	{Sous-vêtements, Pantalons}	4/6
{Pantalons}	4/6	{Caleçon, Baskets}	2/6
{Baskets}	3/6	{Caleçon, Chaussures}	2/6
{Sous-vêtements}	6/6	{Sous-vêtements, Baskets}	3/6
{Chaussures}	3/6	{Sous-vêtements, Chaussures}	3/6
{T-shirt, Pantalons}	2/6		

TAB. 3.3 – Itemsets généralisés fréquents extraits du contexte $\mathcal{V}$ pour *minsupport* = 2/6.

Règle d'association généralisée	Support	Confiance
T-shirt $\rightarrow$ Pantalons	2/6	2/3
Caleçon $\rightarrow$ Pantalons	2/6	2/3
Sous-vêtements $\rightarrow$ Pantalons	4/6	4/6
Pantalons $\rightarrow$ Sous-vêtements	4/6	4/4
Caleçon $\rightarrow$ Baskets	2/6	2/3
Baskets $\rightarrow$ Caleçon	2/6	2/3
Caleçon $\rightarrow$ Chaussures	2/6	2/3
Chaussures $\rightarrow$ Caleçon	2/6	2/3
Baskets $\rightarrow$ Sous-vêtements	3/6	3/3
Chaussures $\rightarrow$ Sous-Vêtements	3/6	3/3

TAB. 3.4 – Règles d'association généralisées valides dans le contexte $\mathcal{V}$ pour *minsupport* = 2/6 et *minconfiance* = 2/3.

L'ensemble FG des itemsets généralisés fréquents dont la taille est $|FG| \leq 2^{\mathcal{I}'}$ est un sur-ensemble de l'ensemble F des itemsets fréquents dont la taille est $|F| \leq 2^{\mathcal{I}}$. L'ensemble des règles d'association généralisées valides $\mathcal{ARG}$, générées à partir de FG, est donc un sur-ensemble de l'ensemble des règles d'association valides $\mathcal{AR}$.

Toutefois, il est possible de supprimer certaines règles d'association généralisées valides lorsque celles-ci sont résumées par une règle plus générale. Une règle d'association généralisée $r' : A' \rightarrow C$ est plus générale qu'une règle $r : A \rightarrow C$ si les items de A' sont des ancêtres des items de A. La règle r est résumée par la règle r' s'il est possible de déduire r, son support et sa confiance à partir de la règle r' et des supports des itemsets A et A'. Dans ce cas la règle r est redondante par rapport à la règle r'.

Afin d'identifier les règles d'association généralisées redondantes, l'intérêt d'une règle r est évalué en comparant son support et sa confiance réels avec le support et la confiance attendus pour r en fonction des règles r' plus générales que r et des support des antécédents de r et r'. Considérons par exemple deux règles "Lait $\rightarrow$ Céréales (support 10%, confiance 50%)" et "Lait écrémé $\rightarrow$ Céréales (support 5%, confiance 50%)". Si l'item "Lait écrémé" est un descendant de l'item "Lait" tel que la moitié des objets contenant un descendant de "Lait" contiennent "Lait écrémé", la règle "Lait écrémé $\rightarrow$ Céréales" est redondante car elle ne convoie aucune information supplémentaire et est moins générale que la règle "Lait $\rightarrow$ Céréales". Son support est exactement celui attendu en considérant les supports des items "Lait" et "Lait écrémé" et de la règle "Lait $\rightarrow$ Céréales". L'approche proposée dans [SA95] consiste à générer les règles d'association généralisées dont le support réel est au moins R fois celui attendu en considérant les supports des items ancêtres et des règles plus générales. La valeur R est une valeur définie par l'utilisateur en même temps que les seuils minimaux de support et confiance. Pour $R = 0$, toutes les règles d'association généralisées sont générées.

Exemple 3.4 Pour $R = 2$, les règles d'association généralisées valides dans le contexte $\mathcal{V}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance de 2/3 sont présentées dans la table 3.5. Les règles "T-shirt $\rightarrow$ Pantalons (support 2/6, confiance 4/6)" et "Caleçon $\rightarrow$ Pantalons (support 2/6, confiance 4/6)" ne sont pas générées car leur support n'est pas R fois celui attendu en considérant la règles "Sous-vêtements $\rightarrow$ Pantalons (support 2/6, confiance 4/6)" dont la confiance est identique. Le support des deux premières règles est exactement celui attendu en considérant le support de la troisième car les occurrences de l'item "Sous-vêtements" correspondent pour la moitié des cas à l'item "T-shirt" et pour l'autre moitié à l'item "Caleçon".

Règle d'association généralisées	Support	Confiance
Sous-vêtements → Pantalons	4/6	4/6
Pantalons → Sous-vêtements	4/6	4/4
Caleçon → Baskets	2/6	2/3
Caleçon → Chaussures	2/6	2/3
Chaussures → Caleçon	2/6	2/3
Chaussures → Sous-Vêtements	3/6	3/3

TAB. 3.5 – Règles d'association généralisées valides dans le contexte $\mathcal{V}$ pour *minsupport* = 2/6, *minconfiance* = 2/3 et $R = 2$.

Trois algorithmes d'extraction des itemsets généralisés fréquents nommés Basic, Cumulate et EstMerge ont été proposés dans [SA95]. Pour l'algorithme Basic, les objets du contexte d'extraction sont remplacés par des objets « étendus » contenant tous les items de l'objet originel ainsi que les items ancêtres de ces items. À partir de ces objets étendus, une version modifiée de l'algorithme Apriori extrait les itemsets généralisés fréquents. Dans l'algorithme Cumulate, une liste des items ancêtres de chaque items de la taxonomie est construite et les items les moins généraux sont remplacés en cours de processus par leur item *père* dans les itemsets déterminés infréquents après un balayage du contexte. Dans l'algorithme EstMerge, les itemsets généralisés fréquents sont déterminés du haut vers le bas dans la taxonomie. L'algorithme détermine les itemsets généralisés fréquents constitués des items les plus généraux et substitue successivement chaque item dans ces derniers par ses items *fils* jusqu'à ce que l'itemset produit soit infréquent.

3.3.2 Utilisation de mesures statistiques

L'utilisation de mesures statistiques autres que la confiance afin de déterminer des relations entre items à fait l'objet de plusieurs études [AY98, BMUT97, BMS97, PS91, SBMU98, SBM98]. Soit $P(l)$ la probabilité d'occurrence d'un itemset $l \subseteq \mathcal{I}$ dans un objet $o \in \mathcal{O}$ du contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$. La quantification de la précision, et donc de « l'intérêt », d'une règle de la forme $A \rightarrow C$ est en général définie en fonction de $P(A)$, $P(C)$ et $P(A \wedge C)$ [PS91]. La confiance d'une règle

d'association $r : l_1 \rightarrow l_2$ est définie par :

$$\textit{confiance}(r) = \frac{P(l_1 \wedge l_2)}{P(l_1)}.$$

Une critique de la mesure de confiance est qu'elle ne tient pas compte de la probabilité $P(l_2)$ d'occurrence de la conséquence de la règle. Si par exemple, nous avons $P(l_2) = 90\%$, $P(l_1) = 80\%$ et $P(l_1 \wedge l_2) = 60\%$, la règle $l_1 \rightarrow l_2$ aura un support de 60% et une confiance de 75%. Cette valeur de confiance est élevée alors que la probabilité qu'un objet contienne l_2 sachant qu'il contient l_1 (75%) est inférieure à la probabilité globale qu'il contienne l_2 (90%). Toutefois, la confiance des règles d'association vérifiées dans 100% des cas est la valeur maximale de confiance possible qui est 1. Cette propriété est importante car elle permet de distinguer ces règles d'association des autres règles. Les mesures statistiques autres que la confiance ont été utilisées afin d'extraire des ensembles constitués d'items dont l'occurrence est corrélée, des règles de dépendances ou des règles causales entre itemsets fréquents. La nature des itemsets et des règles extraites dépend de la mesure statistique utilisée et de l'utilisation de cette mesure dans l'heuristique d'extraction.

Une mesure appelée *intérêt* a été étudiée dans [BMUT97, SBM98]. Cette mesure spécifie le degré de dépendance entre deux itemsets antécédent et conséquence d'une règle d'association en prenant en considération la probabilité d'occurrence de l'itemset conséquence de la règle. Elle est basée sur l'observation que pour deux itemsets l_1 et l_2 sont indépendants statistiquement si $P(l_1 \wedge l_2) = P(l_1)P(l_2)$. L'intérêt d'une règle d'association $r : l_1 \rightarrow l_2$ est défini par :

$$\textit{intérêt}(r) = \frac{P(l_1 \wedge l_2)}{P(l_1)P(l_2)}.$$

Il s'agit donc d'une mesure de déviation de l'indépendance entre l'occurrence de l_1 et l'occurrence de l_2 dans les objets du contexte. Le numérateur est la probabilité réelle d'occurrence des deux itemsets et le dénominateur indique quelle serait la valeur de cette probabilité si les deux itemsets étaient totalement indépendants. Si les deux itemsets sont indépendants, la valeur d'intérêt obtenue sera égale à 1. Une valeur inférieure à 1 indique l'existence d'une dépendance négative et une valeur supérieure à 1 indique l'existence d'une dépendance positive entre les deux itemsets. La mesure d'intérêt a été utilisée à la place de la confiance par Brin et al. [BMUT97] afin de générer les règles d'association à partir de l'ensemble des itemsets fréquents.

Les règles d'association générées sont celles dont l'intérêt est supérieur à 1. Cette mesure pose toutefois deux problèmes. Le premier est que pour les règles vérifiées dans 100% des cas ne possèdent pas nécessairement la mesure d'intérêt maximale. Considérons par exemple deux itemsets l_1 et l_2 parfaitement dépendants, c'est à dire tels que $\text{support}(l_1) = \text{support}(l_2) = \text{support}(l_1 \cup l_2) = 0{,}9$. La mesure d'intérêt pour ces itemsets sera $0{,}9/(0{,}9 \times 0{,}9) = 1{,}11$ qui est une valeur proche de la valeur d'indépendance des itemsets qui est 1. Le second est que la valeur d'intérêt d'une règle spécifie la probabilité de co-occurrence des itemsets dans les objets du contexte, elle ne définie pas l'implication entre les itemsets car elle est symétrique [BMUT97].

Une mesure appelée *conviction* a été définie par Brin et al. dans [BMUT97]. Elle permet de mesurer la déviation de la dépendance entre la présence de l'antécédent et l'absence de la conséquence d'une règle dans les objets. Cette mesure est basée sur l'observation qu'une règle $r : l_1 \rightarrow l_2$ peut être exprimée en logique par $\neg(l_1 \wedge \neg l_2)$. Si la présence de l_1 et l'absence de l_2 dans un objet sont statistiquement sans relation, alors $P(l_1 \wedge \neg l_2) = P(l_1)P(\neg l_2)$. La conviction de la règle r est définie comme suit :

$$conviction(r) = \frac{P(l_1)P(\neg l_2)}{P(l_1 \wedge \neg l_2)}.$$

La fraction est inversée afin de tenir compte de la négation externe de l'expression logique de la règle. Le dénominateur est la probabilité réelle de la présence de l_1 et l'absence de l_2 dans un objet et le numérateur indique quelle serait la valeur de cette probabilité si la présence de l_1 et l'absence de l_2 étaient sans relation. Si les deux événements sont sans relation, la conviction obtenue sera égale à 1 et si les événements sont liés, la valeur obtenue sera supérieure à 1. La dépendance entre les deux itemsets est d'autant plus importante que la valeur est éloignée de 1 : pour une règle $l_1 \rightarrow l_2$ est vérifiée dans 100% des cas, la conviction sera égale à la valeur maximale possible qui est ∞. La conviction est une mesure d'implication entre itemsets car elle est directionnelle et prend en compte $P(l_1)$ ainsi que $P(l_2)$.

La mesure du χ^2 spécifie le degré de dépendance entre différents items d'un itemset en comparant la distribution réelle de leur occurrence avec la distribution attendue de leur occurrence sous l'assomption d'une complète indépendance et d'une distribution normale de l'occurrence de chaque item. L'utilisation du χ^2 afin d'extraire des itemsets fréquents dont le degré de dépendance des items est supérieur ou égal à un seuil donné a été étudié dans [BMS97, SBMU98, SBM98]. Soit $R = \{i_1, \overline{i_2}\} \times \ldots \times \{i_k, \overline{i_k}\}$ le produit cartésien des ensembles d'événements corres-

pondant à la présence et l'absence des items $i_1, \ldots, i_k$ dans un objet. Un élément $r = r_1 \ldots r_k \in R$ correspond à une instance de toutes les variables, c'est à dire un itemset sous-ensemble de l'itemset $\{i_1 \ldots i_k\}$. R est fréquemment représenté comme une table de dimension k appelée *table de contingence* et un élément $r \in R$ correspond alors une *cellule* de la table. La valeur de la cellule r est le nombre d'objets du contexte correspondant à l'instance r noté $O(r)$. Afin de déterminer la dépendance d'une instance r, il faut déterminer si la valeur $O(r)$ de la cellule r diffère suffisamment de la valeur attendue $E[r]$. Soit $n = |\mathcal{O}|$ le nombre total d'objets du contexte. La valeur attendue est calculée sous l'assomption d'indépendance. Nous avons donc pour un item i_j, $E[i_j] = O(i_j)$ et $E[\overline{i_j}] = n - O(i_j)$ et pour une instance r, $E[r] = n \times E[r_1]/n \times \ldots \times E[r_k]/n$. La mesure du χ^2 est définie par :

$$\chi^2 = \sum_{r \in R} \frac{(O(r) - E[r])^2}{E[r]}.$$

La valeur obtenue est la déviation normalisée par rapport à la valeur attendue. Elle spécifie si les k items de l'itemset $\{i_1 \ldots i_k\}$ sont dépendants deux à deux. Si tous les items de l'itemset sont indépendants, la valeur du χ^2 est égale à 0 (avec des fluctuations pour $n < \infty$). Pour une valeur du χ^2 supérieure à un seuil minimal, par exemple 3,84 pour un niveau de signification de 95%, l'assomption d'indépendance est rejetée. Un algorithme d'extraction des itemsets fréquents par niveaux utilisant en plus du support le test du χ^2 est présenté dans [BMS97, SBM98]. Durant une itération k, les k-itemsets fréquents sont extraits du contexte, pour chacun de ces k-itemsets la table de contingence des k items est construite et si la valeur du χ^2 obtenue est inférieure à un seuil minimal défini par l'utilisateur alors l'itemset est supprimé de l'ensemble résultat. Trois algorithmes d'extraction de règles causales appelés CC-path, CU-path et CU-path-heuristic utilisant la mesure du χ^2 sont présentés dans [SBMU98]. Les règles causales extraites sont constituées de trois items et sont de deux types : Les règles causales CCC sont construites entre trois items i_1, i_2 et i_3 dépendants deux à deux tels que i_1 et i_3 deviennent indépendants quand ils sont conditionnés sur i_2. Elles sont de la forme : $i_1 \leftarrow i_2 \rightarrow i_3$ ou $i_1 \rightarrow i_2 \rightarrow i_3$ ou $i_1 \leftarrow i_2 \leftarrow i_3$. Les règles causales CCU sont construites entre trois items i_1, i_2 et i_3 tels que i_1 et i_2 ainsi que i_1 et i_3 sont dépendants, i_2 et i_3 sont indépendants mais deviennent dépendants lorsque ils sont conditionnés sur i_1. Elles sont de la forme $i_2 \wedge i_3 \rightarrow i_1$.

Une mesure appelée *collective strength* a été définie par Aggarwal et al. dans [AY98]. Cette mesure spécifie pour un itemset l une valeur comprise entre 0 et ∞ indiquant la corrélation entre les items de l. Une valeur de 0 indique une absence totale de corrélation alors qu'une valeur de ∞ indique une corrélation parfaite entre les items. La valeur 1 correspond à un itemset dont l'occurrence réelle dans le contexte est celle attendue sous l'assomption d'indépendance statistique des items. Un itemset l est en *violation* d'un objet si certains items de l sont contenus dans l'objet mais pas les autres. Le *taux de violation* d'un itemset l noté $v(l)$ est la proportion d'objets du contexte pour lesquels l est en violation. La valeur $1-v(l)$ est la proportion d'objets du contexte contenant soit tous les items de l, soit aucun des items de l. Soit $E[v(l)]$ (resp. $1-E[v(l)]$) la valeur attendue de $v(l)$ (resp. $1-v(l)$) sous l'assomption d'indépendance statistique des items constituant l. La mesure de collective strength associée à un itemset l est définie par :

$$\textit{collective strength}(l) = \frac{1 - v(l)}{1 - E[v(l)]} \times \frac{E[v(l)]}{v(l)}.$$

Cette mesure a été utilisée par Aggarwal et al. [AY98] afin d'extraire des *itemsets fréquents fortement collectifs* en modifiant l'algorithme Apriori d'extraction des itemsets fréquents par niveaux. Un itemset fréquent l est fortement collectif si la mesure *collective strength*(l) est supérieure ou égale à un seuil minimal *minstrength* défini par l'utilisateur et pour chaque sous-ensemble $l' \subset l$, la mesure *collective strength*(l') est également supérieure ou égale à *minstrength*. Cet algorithme ne prend pas en considération la notion de support des itemsets qui permet de s'assurer qu'un itemset fortement collectif concerne suffisamment d'objets du contexte pour être utile à l'utilisateur [AY98]. Aggarwal et al. proposent donc de supprimer les itemsets fortement collectifs dont le support est inférieur au seuil *minsupport* lors d'une phase de post-traitement. La mesure de *collective strength* pose un problème identique à la mesure d'*intérêt* : elle spécifie la corrélation entre les items des itemsets mais ne défini pas l'implication entre les items puisque elle est symétrique.

3.3.3 Mesures de déviation

Les mesures de déviation sont des mesures de distance entre règles d'association définies en utilisant les supports et les confiances des règles. Elles sont utilisées de deux manières distinctes afin de supprimer certaines règles de l'ensemble des règles

d'association extraites. La première consiste à identifier les règles d'association fortement semblables, caractérisées par une faible distance, et classer ou supprimer certaines de ces règles en fonction des mesures de déviation qui leurs sont associées [BAG99, DL98, TKR+95]. La seconde permet d'identifier les règles d'association qui sont inattendues pour l'utilisateur et qui apportent donc une connaissance importante car nouvelle [Hec96, PSM94, ST95, ST96]. Les connaissances de l'utilisateur sont représentées en utilisant des modèles probabilistes auxquels sont confrontés les règles d'association extraites. La déviation d'une règle correspond à la différence entre la valeur attendue pour la règle dans le modèle probabiliste et la valeur réelle pour la règle dans le contexte.

Une mesure de déviation basée sur la confiance des règles d'association a été proposée dans [DL98]. Cette mesure est calculée pour chaque règle d'association r parmi un ensemble R de règles extraites. Nous notons $moyenne_{conf}(R)$ la valeur moyenne et $deviation_{conf}(R)$ la déviation standard des confiances des règles de l'ensemble R. La mesure de distance associée à une règle d'association $r : l_1 \rightarrow l_2$ dans l'ensemble R est :

$$distance_{std}(r, R) = (confiance(r) - moyenne_{conf}(R)) - deviation_{conf}(R).$$

L'approche proposée dans [DL98] consiste a fractionner l'ensemble $\mathcal{AR}$ des règles d'association valides extraites en plusieurs sous-ensembles et supprimer dans chacun de ces sous-ensemble R les règles r dont la valeur $distance_{std}(r, R)$ est inférieure à un seuil minimal défini par l'utilisateur.

Dans [BAG99], une mesure de déviation est définie en utilisant la notion de *sous-règles* d'une règle d'association. Une règle r' est une sous-règle[2] d'une règle r si les conséquences de r et r' sont identiques et l'antécédent de r' est un sous-ensemble de l'antécédent de r. Formellement, étant donné l'ensemble $\mathcal{AR}$ de règles d'association valides, l'ensemble $SR(r)$ des sous-règles d'une règle d'association $r : A \rightarrow C$ de $\mathcal{AR}$ est défini par $SR(r) = \{r' : A' \rightarrow C \mid r' \in \mathcal{AR} \ \wedge \ A' \subset A\}$. La mesure de déviation est calculée pour chaque règle d'association $r \in \mathcal{AR}$ en fonction des confiances de ses sous-règles. Cette mesure, appelée *improvement*, est définie par :

$$improvement(r) = min_{r' \in SR(r)}(confiance(r) - confiance(r')).$$

[2]Dualement, la règle r est appelée *sur-règle* de la règle r'.

Elle permet de « quantifier » l'information supplémentaire convoyée par la règle r par rapport à ses sous-règles. Si la valeur *improvement*(r) est positive, la suppression d'un ou plusieurs items de l'antécédent de r entraîne une diminution de la confiance de la règle au moins égale à *improvement*(r). Pour une valeur élevée, chaque item de l'antécédent contribue donc de manière importante à la prédiction que représente la règle. Si la valeur *improvement*(r) est inférieure ou égale à 0, la règle r n'est pas informative car elle est peut être simplifiée par une de ses sous-règle r' au moins aussi informative et qui s'applique à une population plus importante du fait de la condition sur leurs antécédents $A' \subset A$. Un algorithme nommé Dense-Miner, qui est une extension de l'algorithme Max-Miner, permettant de générer les règles d'association de la forme $r : l_1 \rightarrow C$ valides dont la mesure *improvement*(r) est supérieure ou égale à un seuil minimal *minimprovement* est présenté dans [BAG99]. L'itemset conséquence $C \subset \mathcal{I}$ des règles extraites est spécifiée en début de processus. Cette contrainte sur la conséquence des règles fait partie de l'approche orientée utilisateur par *contrainte sur les items* présentée dans la section 3.4.3.

Une mesure de déviation spécifiant une distance entre deux règles d'association possédant la même conséquence a été proposée dans [TKR+95]. La distance entre les deux règles est définie en fonction des ensembles d'objets qui vérifient chacune des deux règles. Soit $O(l)$ le nombre d'objets du contexte d'extraction contenant l'itemset l. Pour deux règles d'association $r : A \rightarrow C$ et $r' : A' \rightarrow C$ la distance entre r et r' est :

$$distance_{sem}(r, r') = O(A \cup C) + O(A' \cup C) - 2 \times O(A \cup A' \cup C).$$

La valeur obtenue correspond au nombre d'objets du contexte qui vérifient l'une des deux règles mais pas l'autre. Cette mesure a été proposée dans [TKR+95] afin de faciliter la visualisation de l'ensemble de règles d'association extraites en formant des groupes ou clusters de règles qui concernent les mêmes objets du contexte. Les clusters sont construits en minimisant la distance entre les règles de chaque cluster.

Les réseaux Bayésiens sont les modèles probabilistes les plus utilisés afin de définir les connaissances de l'utilisateur [Hec96, ST95, ST96]. Dans l'interprétation Bayésienne, la probabilité qu'un événement e se produise selon l'état ξ de connaissance de l'utilisateur est notée $P(e, \xi)$. La probabilité que l'événement e_2 se produise sachant que l'événement e_1 est vérifié et selon l'état de connaissance ξ est notée $P(e_2|e_1, \xi)$. Ces probabilités définies par l'utilisateur sont utilisées afin de construire un réseau

Bayésien B pour l'ensemble $\mathcal{I}$ représentant la distribution des probabilité $P(\mathcal{I}, \xi)$. Étant donnée une nouvelle information r, le degré de probabilité d'un événement e sachant r noté $P(e|r, \xi)$ est calculé en utilisant la règle Bayésienne :

$$P(e|r, \xi) = \frac{P(r|e, \xi)P(e, \xi)}{P(r|e, \xi)P(e, \xi) + P(r|\neg e, \xi)P(\neg e, \xi)}.$$

La mesure de déviation d'une nouvelle information r par rapport au réseau B est :

$$déviation_{Bay}(r, B) = \sum_{e \in B} \frac{P(e|r, \xi) - P(e, \xi)}{P(e, \xi)}.$$

Cette valeur mesure combien la nouvelle information modifie le système de connaissances. Les informations pour lesquelles cette valeur est la plus élevée sont les plus inattendues pour l'utilisateur. L'application de cette approche pose plusieurs problèmes importants. Le premier est la nécessité pour l'utilisateur de concevoir le système de connaissances, ce qui dans de nombreux cas se révèle très complexe. Le second concerne les temps des calculs des mesures de déviations qui sont très importants car ils requièrent de très nombreuses opérations puisqu'il faut calculer la somme sur tous les événements de B.

3.3.4 Couverture structurelle

Dans [TKR+95], l'ensemble des règles d'association valides $\mathcal{AR}$ est extrait et une *couverture structurelle* pour les règles d'association est construite en supprimant de cette ensemble les règles redondantes d'un point de vue syntaxique. Pour cela, des sous-ensembles de l'ensemble $\mathcal{AR}$ sont construits et chaque sous-ensemble est examiné successivement. Un sous-ensemble est constitué de toutes les règles valides qui possèdent le même itemset conséquence. La sélection des règles supprimées à l'intérieur d'un sous-ensemble est basée sur l'observation que pour deux règles r : $A \rightarrow C$ et r' : $A' \rightarrow C$ telles que $A' \subset A$ (r' est une sous-règle de r), l'ensemble des objets vérifiant la règle r' est un sur-ensemble de l'ensemble des objets vérifiant la règle r. En effet, puisque $(A' \cup C) \subset (A \cup C)$, l'ensemble des objets contenants l'itemset $A' \cup C$ et un sur-ensemble de l'ensemble des objets contenants l'itemset $A \cup C$. La couverture structurelle Δ d'un ensemble $\mathcal{AR}$ de règles d'association est définie par :

$$\Delta = \{A \rightarrow C \in \mathcal{AR} \mid \nexists A' \rightarrow C \in \mathcal{AR} \text{ telle que } A' \subset A\}.$$

En utilisant la notion de *sur-règle* (voir section 3.3.3), l'ensemble Δ contient les règles d'association de $\mathcal{AR}$ qui ne possèdent aucune sur-règle dans l'ensemble $\mathcal{AR}$. La couverture structurelle ainsi définie contient les règles d'association les plus générales d'un point de vue syntaxique de l'ensemble $\mathcal{AR}$.

Cette méthode si elle permet de réduire efficacement le nombre de règles d'association extraites ne tient toutefois pas compte de la confiance des règles d'association. Pour qu'aucune information ne soit perdue, le contexte d'extraction des règles doit être uniforme, c'est à dire que pour une règle $r : A \rightarrow C$, il ne doit pas exister de sur-règle de r avec une confiance inférieure. Ceci pose un problème car la confiance d'une règle r ne peut être déduite de la confiance de ses sur-règles ou sous-règles. Ainsi, si par exemple nous avons deux règles d'association valides "Lait $\rightarrow$ Céréales (confiance 80%)" et "Lait Café $\rightarrow$ Céréales (confiance 40%)", la première règle qui est une sous-règle de la seconde sera supprimée alors que du fait de sa confiance bien plus élevée elle est plus informative pour l'utilisateur que la seconde.

Un algorithme de génération de couvertures structurelles appelé RuleCover est présenté dans [TKR+95]. Afin de générer la couverture structurelle, cet algorithme requiert qu'à chaque règle d'association de $\mathcal{AR}$ soit associée la liste des objets vérifiant la règle, c'est à dire la liste des objets contenant l'itemset union de l'antécédent et de la conséquence de la règle. Cette contrainte nécessite soit l'extension des algorithmes d'extraction des itemsets fréquents, soit un traitement supplémentaire qui requiert des balayages du contexte, ce qui constitue dans les deux cas un surcoût non négligeable en temps d'exécution et en espace mémoire.

3.3.5 Discussion

Les règles d'association généralisées étendent la définition des règles d'association entre itemsets proposée dans [AS94]. Afin de générer ces règles, une taxonomie des items du contexte d'extraction est utilisée et cette méthode n'est applicable que si une telle taxonomie est présente. Les règles d'association généralisées sont générées à partir des itemsets généralisés fréquents, construits à partir de l'ensemble $\mathcal{I}' \supset \mathcal{I}$ des items de la taxonomie, qui constituent un sur-ensemble de l'ensemble des itemsets fréquents. L'espace de recherche des itemsets généralisés fréquents est donc plus grand que l'espace de recherche des itemsets fréquents, ce qui entraîne une augmentation importante du coût de l'extraction des itemsets nécessaires à la

génération des règles. De plus, l'ensemble des règles d'association généralisées valides est un sur-ensemble de l'ensemble des règles d'association valides. Afin de diminuer le nombre de règles d'association généralisées générées, la valeur de R utilisée doit être élevée (nettement supérieure à 1), ce qui entraîne la suppression de règles non redondantes et donc une perte d'information.

L'utilisation de mesures statistiques autres que la confiance permet dans de nombreux cas d'améliorer la qualité de la mesure de précision des règles d'association. Toutefois, les mesures d'*intérêt* et de *collective strenght* ne définissent pas des implications entre itemsets mais des probabilités de co-occurrence des items dans les itemsets. Ceci ne correspond aux besoins définis par les règles d'association pour lesquels la probabilité de co-occurrence des items dans les objets est conditionnée sur certains items (ceux de l'antécédent de la règle). De plus, le calcul de l'*intérêt* d'une règle $l_1 \rightarrow l_2$ est plus coûteux que le calcul de la confiance de cette règle. Il nécessite la recherche dans la structure de données des supports des itemsets l_1, l_2 et $l_1 \cup l_2$ alors que le support de l'itemset l_2 n'est pas utilisé pour calculer la confiance de la règle. Pour calculer la *collective strenght* d'un itemset, il faut calculer le taux de violation de cet itemset, ce qui représente un surcoût non négligeable en temps d'exécution par rapport à l'extraction des itemsets fréquents.

Pour calculer la mesure de *conviction* d'une règle $l_1 \rightarrow l_2$, les supports des événements $\neg l_2$ et $l_1 \wedge \neg l_2$ doivent être connus. Si le support de l'événement $\neg l_2$ est facilement calculé à partir du support des itemsets fréquents puisque support($\neg l_2$) = 1 - support(l_2), le support de l'événement $l_1 \wedge \neg l_2$ ne peut pas l'être. Le calcul du support des événements de ce type durant l'extraction des itemsets fréquents n'est pas réalisable car le nombre d'événements à considérer est trop important : il est exponentiel dans le nombre d'itemsets fréquents. Ce calcul doit donc faire l'objet d'un post-traitement qui requiert des balayages du contexte et augmente donc de manière non négligeable les temps d'exécution. L'utilisation de la mesure du χ^2 pose deux principaux problèmes. Le premier est le nombre d'opérations nécessaires au calcul des mesures du χ^2 pour chaque itemset fréquent qui est très important et pose des problèmes d'efficacité lorsque les règles sont extraites à partir de grandes bases de données [AY98]. Le second concerne la précision de la mesure qui se dégrade considérablement si les valeurs attendues $E[r]$ sont faibles. La précision n'est garantie que si toutes les cellules de la table de contingence ont une valeur supérieure à 1 et si 80% des cellules ont une valeur supérieure à 5. Dans le cas de règles d'association,

ces conditions ne sont que rarement respectées et la mesure du χ^2 obtenue dans de tels cas est une approximation dont la précision est variable [BMS97, SBMU98]. La définition d'une mesure précise lorsque ces conditions ne sont pas respectées est un problème de recherche ouvert dans la communauté des statistiques.

Les mesures de déviation permettent d'identifier parmi l'ensemble de règles d'association valides extraites les règles qui représentent une variation dans la distribution des confiances des règles par rapport à son voisinage. Ces mesures sont utilisées afin de réduire l'ensemble des règles d'association après leur génération, ce qui nécessite un post-traitement des règles entraînant des temps d'exécution supplémentaires. La mesure $distance_{std}$ est soumise au problème du coût du calcul de la déviation standard des sous-ensembles de règles qui nécessite de nombreuses opérations. L'utilisation de la mesure d'*improvement* pour supprimer les règles redondantes dont la mesure est inférieure à 0 ne réduit que faiblement la taille de l'ensemble résultat. En imposant un seuil minimal *minimprovement* et en supprimant les règles dont l'improvement est inférieur à cette valeur il y a perte d'information car les règles dont la mesure est supérieure à 0 ne sont pas redondantes. La mesure $distance_{sem}$ ne permet pas de réduire l'ensemble des règles extraites. Elle défini seulement un critère de regroupement des règles afin d'en faciliter la visualisation. L'utilisation de réseaux Bayésiens pose deux problèmes. D'une part, le problème de la complexité de la définition du modèle probabiliste par l'utilisateur du fait du nombre d'items du contexte d'extraction qui est en général très important. D'autre part, le problème du coût des calculs des $déviation_{Bay}$ pour les règles extraites qui est très importants.

La définition de la couverture structurelle pour les règles d'association ne tient pas compte de la confiance des règles, ce qui entraîne une perte d'information qui peut dans de nombreux cas être importante. De plus, l'algorithme proposé pour construire cette couverture nécessite pour chaque règle d'association la liste des objets contenant l'itemset union de l'antécédent et de la conséquence de la règle. Afin de construire ces listes, lorsque les règles d'association valides sont extraites, un traitement supplémentaire durant lequel des balayages du contexte seront réalisés est nécessaire. Ce traitement constitue un surcoût non négligeable en temps d'exécution et en espace mémoire pour l'extraction des règles d'association.

Nicolas Pasquier, LIMOS

3.4 Réduction de l'ensemble des règles d'association : approches orientées utilisateur

Les approches orientées utilisateur pour la réduction de l'ensemble des règles d'association requièrent l'intervention de l'utilisateur afin de définir des critères de sélection des règles qui figureront dans l'ensemble résultat. Nous distinguons trois catégories différentes parmi ces approches. La première est l'utilisation d'expressions régulières appelées *templates* utilisées afin de filtrer l'ensemble de règles d'association valides extraites précédemment. L'approche par templates est présentée dans la section 3.4.1. La seconde est l'utilisation d'un opérateur appelé MINE RULE qui est une extension du langage de requête SQL. Les critères de sélection correspondent aux paramètres de cet opérateur qui extrait les règles directement depuis les tuples des tables de la base de données. Cet opérateur et les extensions de ce dernier sont présentés dans la section 3.4.2. Dans la troisième, les critères de sélection sont appelés *contraintes sur les items* et sont utilisés afin d'extraire seulement les itemsets fréquents permettant de générer les règles vérifiant ces contraintes. Les approches entrant dans cette catégorie sont décrites dans la section 3.4.3.

3.4.1 Templates

Dans [KMR+94], les templates sont définis comme des expressions régulières spécifiant des critères généraux de sélection d'un sous-ensemble de règles d'association qui sera présenté à l'utilisateur. Ce sous-ensemble est construit à partir de l'ensemble des règles d'association valides, en conservant les règles qui vérifient les critères spécifiés par les templates parmi cet ensemble. Ces critères stipulent quels items doivent apparaître dans l'antécédent et la conséquence des règles du sous-ensemble. Cette approche a été développée sur le modèle des règles d'association définies dans [AIS93b] dont la conséquence contient un seul item. Elle peut toutefois facilement être étendue pour prendre en considération les règles dont la conséquence contient plusieurs items.

Lors de la définition des templates, une hiérarchie de classes ou taxonomie des items peut éventuellement être prise en considération. Une classe $C(i)$ de la taxonomie est l'ensemble des items feuilles de la taxonomie qui sont des descendant de l'item i. Dans la taxonomie $\mathcal{T}$ associée au contexte $\mathcal{V}$ par exemple, la classe C(Chaussures)

correspond à l'ensemble {Mocassins, Baskets}. Les templates permettent alors de spécifier à quelles classes doivent appartenir les items de l'antécédent et la conséquence des règles du sous-ensemble. La définition des templates présentée dans la suite prend en considération une telle hiérarchie de classes d'items. Les templates sont des expressions de la forme :

$$X_1 \wedge \ldots \wedge X_j \rightarrow X_{j+1},$$

dans laquelle chaque X_h est un item $i \in \mathcal{I}$, une classe d'items $C(i)$ ou bien une expression $C(i)+$ ou $C(i)*$. Les expression $C(i)+$ ou $C(i)*$ signifient une à plusieurs et zéro à plusieurs items de la classe $C(i)$ respectivement. Deux types de templates peuvent être définis, les *templates inclusifs* et les *templates restrictifs*. Les premiers permettent à l'utilisateur de définir explicitement quelles règles sont intéressantes et les seconds, quelles règles ne le sont pas.

Une règle d'association $l_1 \rightarrow l_2$ vérifie un template $X_1 \wedge \ldots \wedge X_j \rightarrow X_{j+1}$ si l_2 vérifie l'expression X_{j+1} et pour chaque $X_h \in (X_1 \wedge \ldots \wedge X_j)$, l_1 vérifie l'expression X_h. Si une expression X est un item i, un itemset l vérifie X si l contient i. Si une expression X est une classe d'items $C(i)$, un itemset l vérifie X si l contient un et un seul descendant de l'item i. Si une expression X est de la forme $C(i)+$ alors l vérifie l'expression X si l contient ou un ou plusieurs descendants de l'item i. Un itemset l vérifie toujours une expression $X = C(i)*$ puisqu'elle spécifie que l'itemset peut contenir de 0 à plusieurs items descendants de i. Les règles qui seront sélectionnées parmi l'ensemble des règles extraites sont celles qui vérifient au moins un template inclusif et ne vérifient aucun des templates restrictifs.

Exemple 3.5 Les règles d'association valides dans le contexte $\mathcal{V}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance de 1/2 sont présentées dans la table 3.6. Supposons maintenant que l'utilisateur cherche à savoir quels articles sont le plus souvent achetés par les clients qui achètent des sous-vêtements. Utilisant la taxonomie $\mathcal{T}$ associée au contexte $\mathcal{V}$, il va être possible de sélectionner les règles correspondants à cette recherche d'informations en spécifiant le template "C(Sous-vêtements)+ $\rightarrow$ C(Article)". La classe C(Article) spécifie que la conséquence des règles extraites peut contenir n'importe lequel des items feuilles de la taxonomie puisque C(Article) = {T-shirt, Caleçon, Pantalon, Mocassins, Baskets}. Les règles d'association valides correspondant à ce template sont les règles "T-shirt $\rightarrow$ Pantalons (2/6, 2/3)", "Caleçon $\rightarrow$ Pantalons (2/6, 2/3)" et "Caleçon $\rightarrow$ Baskets (2/6,

Règle d'association	Support	Confiance
T-shirt → Pantalons	2/6	2/3
Pantalons → T-shirt	2/6	2/4
Caleçon → Pantalons	2/6	2/3
Pantalons → Caleçon	2/6	2/4
Caleçon → Baskets	2/6	2/3
Baskets → Caleçon	2/6	2/3

TAB. 3.6 – Règles d'association valides dans le contexte $\mathcal{V}$ pour $minsupport = 2/6$ et $minconfiance = 1/2$.

2/3)". Ces règles sont les règles d'association valides qui répondent aux besoins de l'utilisateur pour sa recherche d'information.

3.4.2 Opérateur MINE RULE

Dans [MPC96], les critères de selection des règles sont définis par un opérateur nommé MINE RULE. Cet opérateur est une extension du langage de requête SQL implémenté dans les SGBD relationnels. L'extraction des règles d'association est réalisée à partir des tuples des relations de la base de données par une requête qui est une instance de l'opérateur MINE RULE. Les itemsets fréquents sont des ensembles de valeurs d'attributs de tuples extraits des tables (ou requêtes) qui sont les sources de données de la requête. Les règles d'association sont générées à partir de ces itemsets fréquents selon les critères définis par la requête. La syntaxe de l'opérateur MINE RULE est la suivante :

```
MINE RULE <NomTable> AS
SELECT DISTINCT <1|m..1|n> <DescrAntécédant> AS BODY,
                <1|p..1|q> <DescrConséquence> AS HEAD
                [,SUPPORT] [,CONFIANCE]
[WHERE <ClauseWhere1>]
FROM <ListeFrom> [WHERE <ClauseWhere2>]
GROUP BY <ListeAttribut1> [HAVING <ClauseHaving1>]
[CLUSTER BY <ListeAttribut2> [HAVING <ClauseHaving2>]]
EXTRACTING RULE WITH SUPPORT :<ValSupport>,
                     CONFIANCE :<ValConfiance> ;
```

Les expressions entre [et] sont les clauses optionnelles et les expressions entre < et > sont les paramètres de l'opérateur.

La clause FROM défini la liste de tables ou requêtes de la base de données à partir desquelles seront extraits les itemsets. La clause optionnelle WHERE <ClauseWhere2> décrit des critères de sélection des tuples, de ces tables ou requêtes, qui seront utilisés. Les tuples ainsi sélectionnés sont regroupés par valeurs communes pour les attributs de la liste <ListeAttribut1> de la clause GROUP BY. Pour chacun de ces groupes, les valeurs des attributs de la liste <DescrAntécédant> dans les tuples du groupe constituent un itemset antécédent candidat pour les règles. Les valeurs des attributs de la liste <DescrConséquence> dans les tuples du groupe constituent un itemset conséquence candidat pour les règles. La clause CLUSTER BY permet de former des regroupements de tuples à l'intérieur de chaque groupe, par valeurs communes pour les attributs de la liste <ListeAttribut2>. Les itemsets antécédents et conséquences candidats sont alors constitués par l'union des valeurs des attributs dans les tuples des clusters ainsi formés.

Pour chaque couple d'itemset antécédent candidat l_1 et itemset conséquence candidat l_2, le support et la confiance de la règles d'association $l_1 \rightarrow l_2$ sont testés par rapport au seuil minimaux <ValSupport> et <ValConfiance>. La clause optionnelle WHERE <ClauseWhere1> permet de définir des critères supplémentaires de sélection des règles. Ces critères peuvent concerner les attributs des tuples dont sont issus les items de l'antécédent et la conséquence de la règle ou bien d'autre attributs de ces tuples. Les règles pour lesquelles ces critères ne sont pas respectés sont supprimées de l'ensemble résultat. La sémantique de l'opérateur défini dans [MPC96] permet également de prendre en considération une hiérarchie de classes des valeurs des attributs. Les items constituant les règles extraites sont alors soit des valeurs d'attributs, soit des classes de valeurs d'attributs et les critères de sélection et de regroupement des tuples et des règles peuvent porter sur ces classes de valeurs des attributs.

La définition des requêtes correspondants aux besoins de l'utilisateur par l'opérateur MINE RULE nécessite des connaissances approfondies du language de requête SQL. Afin qu'il soit possible pour un utilisateur non-expert de définir ces requêtes, une extension de l'opérateur MINE RULE a été proposée dans [BP97]. Cette extension peut être vue comme une interface simplifiant l'utilisation de cet opérateur grace à des classes de critères d'extraction précédemment définis. Une requête cor-

respond à une instance d'une de ces classes dans laquelle le nombre de paramètres à définir est réduit et les types des paramètres sont simplifiés.

3.4.3 Contraintes sur les items

Dans l'approche par contraintes sur les items, l'utilisateur défini des contraintes, spécifiant les règles d'association à extraire, qui sont des expressions portant sur l'antécédent, la conséquence ou l'antécédent et la conséquence simultanément des règles. Ces contraintes sont utilisées lors de la phase d'extraction des itemsets fréquents afin de limiter l'espace de recherche de cette phase. Elles sont prises en compte lors de la génération des itemsets candidats afin de considérer seulement les candidats permettant de générer des règles satisfaisant les contraintes.

L'approche proposée dans [SVA97] consiste à limiter l'extraction des règles d'association à un sous-ensemble de règles vérifiant une expression booléenne spécifiant la présence ou l'absence des items dans les règles, sans distinction entre l'antécédent et la conséquence. Cet expression booléenne peut être définie en utilisant une taxonomie des items si elle existe et les éléments de l'expression peuvent alors être également de la forme *ancêtre*($item$) ou *descendant*($item$). Elle est exprimée comme une *forme normale disjonctive* de la forme :

$$D_1 \vee \ldots \vee D_j,$$

dans laquelle chaque élément D_k est de la forme $a_k^1 \wedge \ldots \wedge a_k^p$. Lorsque aucune taxonomie n'est disponible, chaque élément a_k^q est de la forme i ou bien $\neg i$ pour un item $i \in \mathcal{I}$. Lorsque une taxonomie est disponible, un élément a_k^q peut également être de la forme *ancêtre*(i), *descendant*(i), $\neg$*ancêtre*(i) ou $\neg$*descendant*(i). Lors de l'évaluation de l'expression, *descendant*(i) est remplacé par $(i \vee i' \vee i'' \vee \ldots)$ et $\neg$*descendant*(i) par $\neg(i \vee i' \vee i'' \vee \ldots)$ avec $i', i'', \ldots$ qui sont les descendants de i dans la taxonomie. La même opération est réalisée pour *ancêtre*(i) et $\neg$*ancêtre*(i). Étant donnée une expression B définie par l'utilisateur, les règles d'association extraites sont les règles $r : l_1 \rightarrow l_2$ pour $l_1, l_2 \in F$ telles que $l_1 \cup l_2$ satisfait B, $support(l_1 \cup l_2) \geq minsupport$ et $confiance(r) \geq minconfiance$. L'itemset $l_1 \cup l_2$ satisfait une expression B si tous les items de $l_1 \cup l_2$ ont une valeur *vrai* dans B et tous les autres items de B ont une valeur *faux*. L'extraction des règles d'association satisfaisant une expression B peut être divisé en trois phases :

1. Extraire tous les itemsets fréquents satisfaisant l'expression B. Trois approches peuvent être utilisées pour réaliser cette phase. La première consiste à extraire tous les itemsets fréquents et supprimer ceux ne satisfaisant pas l'expression B ; La seconde, à modifier la procédure de génération des candidats afin de générer tous itemsets candidats et calculer le support de ceux satisfaisant B ; La troisième, à modifier la procédure de génération des candidats afin de ne générer que les itemsets candidats satisfaisant B.

2. Déterminer le support de tous les sous-ensembles des itemsets extraits durant la phase 1. En effet, afin de calculer la confiance d'une règle $l_1 \rightarrow l_2$ générée à partir d'un itemset fréquent $l = l_1 \cup l_2$ satisfaisant B, le support de l'itemset l_1 est nécessaire. Cet itemset peut ne pas satisfaire B et n'avoir donc pas été extrait durant la phase 1. Il faut donc déterminer tous les sous-ensembles des itemsets fréquents extraits durant la phase 1 et réaliser un balayage supplémentaire afin de calculer leurs supports.

3. Générer toutes les règles d'association valides à partir des itemsets fréquents extraits durant la phase 1. Durant cette phase, les supports des itemsets calculés durant les phases 1 et 2 sont utilisés pour déterminer la confiance de chaque règle d'association.

L'approche la plus simple pour extraire les règles satisfaisant l'expression B consiste à extraire toutes les règles d'association valides et supprimer ensuite celles ne satisfaisant pas l'expression B. Toutefois, la réduction de l'extraction de l'ensemble des itemsets fréquents à ceux satisfaisant B permet de réduire significativement les temps d'exécution comparé à cette dernière approche. Trois algorithmes qui sont des extensions de l'algorithme Apriori afin de prendre en considération l'expression lors de l'extraction des itemsets fréquents ont été présentés dans [SVA97]. Chacun de ces algorithmes correspond à une des trois approches énumérées dans la description de la phase 1.

Dans [NLHP98], les contraintes spécifiant les règles d'association extraites sont plus générales que dans la précédente approche. Ces contraintes sont des expressions qui peuvent porter sur l'antécédent ou la conséquence des règles seulement, ou bien sur l'antécédent et la conséquence simultanément. L'extraction des règles d'association est réalisée à partir des tuples des relations de la base de données (comme pour l'opérateur MINE RULE) et les itemsets sont des ensembles de valeurs d'attributs

des tuples. Pour une règle d'association $r : l_1 \rightarrow l_2$ générée à partir de l'itemset $l = l_1 \cup l_2$, les diverses types de contraintes qui peuvent être appliquées sont les suivants :

- Contraintes d'occurrence de valeurs de certains attributs dans l_1, l_2 ou l. Par exemple : $\exists i \in l,\ i \in domaine(\text{Article})$ qui spécifie que la règle doit contenir une valeur de l'attribut Article.
- Contraintes de domaines de valeurs des attributs dans l_1, l_2 ou l. Par exemple : $\forall i \in l_2,\ i.\text{Prix} > 100$ qui spécifie que les articles de la conséquence de la règle doivent avoir un prix supérieur à 100.
- Contraintes des opérations d'agrégation sur les attributs de l_1, l_2 ou l. Par exemple : $\forall i \in l_1 \mid i \in domaine(\text{Article}),\ somme(i.\text{Prix}) < 500$ qui spécifie que la somme des prix des articles de l'antécédent de la règle doit être inférieure à 500.
- Contraintes sur les attributs de l_1 et l_2 simultanément. Par exemple : $\forall i \in l_1 \wedge \forall i' \in l_2 \mid domaine(i) \neq domaine(i')$ qui spécifie que les valeurs contenues dans l'antécédent et la conséquence de la règles doivent concerner des attributs différents.

Trois algorithmes, nommés Apriori+, Hybrid et CAP, d'extraction des itemsets fréquents nécessaires à la génération des règles d'association valides vérifiant un ensemble de contraintes $\mathcal{C}$ sont présentés dans [NLHP98]. Ces itemsets sont les itemsets fréquents qui peuvent être l'antécédent, la conséquence ou l'union de l'antécédent et de la conséquence d'une telle règle. Par abus de langage, nous disons que ces itemsets vérifient l'ensemble $\mathcal{C}$ de contraintes. Les algorithmes Apriori+ et Hybrid sont des extensions de l'algorithme Apriori. L'algorithme Apriori+ génère l'ensemble des itemsets fréquents et supprime ensuite ceux qui ne vérifient pas l'ensemble $\mathcal{C}$. Dans l'algorithme Hybrid, la procédure de génération des itemsets candidats est modifiée afin de ne générer que les itemsets candidats qui vérifient $\mathcal{C}$. Pour l'algorithme CAP, les contraintes de l'ensemble $\mathcal{C}$ sont classées en quatre catégories selon leur propriétés de *succincteté* et d'*anti-monotonie*. Les contraintes de chaque catégorie sont ensuite utilisées à diverses étapes de l'extraction afin de minimiser le nombre d'itemsets considérés, ce qui améliore sensiblement les temps d'exécution par rapport aux algorithmes Apriori+ et Hybrid.

Dans [BAG99], une contrainte unique sur la conséquence des règles d'association extraites est définie par l'utilisateur. Cette contrainte est un itemset $C \subset \mathcal{I}$

qui spécifie l'itemset conséquence des règles d'association extraites qui seront de la forme $r : A \rightarrow C$. Afin de générer ces règles et de calculer leurs supports et leurs confiances, tous les couples d'itemsets fréquents A et $A \cup C$ doivent être extraits. L'algorithme Dense-Miner, présenté dans [BAG99], permet d'extraire ces itemsets et de générer les règles $r : A \rightarrow C$ valides. Comme précisé précédemment, cet algorithme prend également en considération la mesure appelée *improvement* associée à chaque règle r (voir section 3.3.3). Si la mesure *improvement*(r) est inférieure à un seuil minimal *minimprovement* défini par l'utilisateur, la règle r est supprimé de l'ensemble résultat.

3.4.4 Discussion

Les approches orientées utilisateurs focalisent la recherche sur un sous-ensemble des règles d'association valides délimité par les spécifications de l'utilisateur. Ainsi, les règles d'association inattendues pour l'utilisateur ne sont pas extraites. Or, ces règles constituent une information importante pour l'utilisateur car elles lui apportent une connaissance nouvelle [DL98, Hec96, PSM94, ST95, ST96].

De plus, les règles d'association redondantes, qui ne convoient aucune information supplémentaire et dont la présence n'est pas souhaitable, ne sont pas supprimées des sous-ensembles de règles extraites. Ces règles nuisent de manière importante à la pertinence et l'utilité du résultat puisque elles représentent dans certains cas une proportion majoritaire des règles extraites, particulièrement pour les contextes d'extraction constitués de données denses ou corrélées.

L'opérateur MINE RULE est défini comme une extension du langage de requête SQL. Lors de l'extraction des règles d'association correspondant à une requête, le SGBD est utilisé afin d'extraire les valeurs des tuples dans la base de données. Ceci ralenti considérablement les temps d'exécution par rapport aux processus d'extraction pour lesquels le contexte d'extraction (relation binaire) est construit lors d'une phase de pré-traitement. Ce pré-traitement permet de plus de supprimer les valeurs inintéressantes dans le jeu de données qui seront considérées par l'opérateur MINE RULE puisque présentent dans les tuples de la base de données. De plus, l'opérateur MINE RULE utilise des opérations d'agrégation pour spécifier la structure des règles extraites, ce qui entraîne des problèmes de performances car ces opérations sont coûteuses en temps d'exécution.

Deuxième partie

Approche par Fermeture de la Connexion de Galois

Chapitre 4

Nouvelle sémantique pour l'extraction de règles d'association

Sommaire

4.1 Introduction

Nous proposons une nouvelle sémantique basée sur la connexion de Galois pour le problème de l'extraction de règles d'association [PBTL98, PBTL99b, PBTL99c]. Utilisant les opérateurs de fermeture de la connexion de Galois, nous définissons les *itemsets fermés*, qui forment le *treillis des itemsets fermés*, et les *itemsets fermés fréquents*. Nous démontrons que l'ensemble des itemsets fermés fréquents constituent un *ensemble générateur*, également appelé *base*, pour l'ensemble des itemsets fréquents. Cela signifie que les itemsets fréquents et leurs supports peuvent être générés à partir des itemsets fermés fréquents et leurs supports sans accéder à la base de données. Nous proposons une nouvelle décomposition du problème de l'extraction de règles d'association consistant à extraire les itemsets fermés fréquents au lieu des itemsets fréquents. Cette décomposition permet d'améliorer les temps de réponse car dans la plupart des cas le nombre d'itemsets fermés fréquents est bien inférieur au nombre d'itemsets fréquents. Utilisant les itemsets fermés fréquents, nous définissons des bases[1] pour les règles d'association. Ces bases, qui sont des sous-ensembles de l'ensemble des règles d'association valides, permettent d'améliorer la pertinence et l'utilité de l'ensemble de règles extraites.

4.2 Connexion de Galois

Dans cette section, nous rappelons les définitions des opérateurs de fermeture, de la connexion de Galois et de la fermeture de la connexion de Galois utilisées par la suite pour définir le treillis des itemsets fermés. Nous définissons également formellement le support des itemsets en utilisant la connexion de Galois.

[1] Nous utilisons dans ce mémoire le terme « base » au sens d'ensemble générateur non redondant. Une base pour un ensemble de règles E est un sous-ensemble de E ne contenant aucune règle redondante et à partir duquel toutes les règles appartenant à E peuvent être déduites.

4.2.1 Opérateurs de fermeture

Soit un ensemble partiellement ordonné $(E, \leq)$. Une application γ de $(E, \leq)$ dans $(E, \leq)$ est un opérateur de fermeture si et seulement si elle possède les trois propriétés suivantes pour tous sous-ensembles $S, S' \subseteq E$:

1. Isotonie : $S \leq S' \Rightarrow \gamma(S) \leq \gamma(S')$,
2. Extensivité : $S \leq \gamma(S)$,
3. Idempotence : $\gamma(\gamma(S)) = \gamma(S)$.

Étant donné un opérateur de fermeture γ sur un ensemble partiellement ordonné $(E, \leq)$, un élément $x \in E$ est un élément *fermé* si l'image de x par l'opérateur de fermeture est lui-même : $\gamma(x) = x$.

4.2.2 Connexion de Galois

Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$. Soit l'application ϕ de l'ensemble des parties de $\mathcal{O}$[2] dans l'ensemble des parties de $\mathcal{I}$ qui associe à un ensemble d'objets $O \subseteq \mathcal{O}$ l'ensemble des items $i \in \mathcal{I}$ communs à tous les objets $o \in O$:

$$\phi : 2^{\mathcal{O}} \rightarrow 2^{\mathcal{I}}$$
$$\phi(O) = \{i \in \mathcal{I} \mid \forall o \in O \text{ nous avons } (o, i) \in \mathcal{R}\}$$

Soit l'application ψ de l'ensemble des parties de $\mathcal{I}$ dans l'ensemble des parties de $\mathcal{O}$ qui associe à tout ensemble d'items (itemset) $I \subseteq \mathcal{I}$ l'ensemble des objets $o \in \mathcal{O}$ contenant tous les items $i \in I$[3] :

$$\psi : 2^{\mathcal{I}} \rightarrow 2^{\mathcal{O}}$$
$$\psi(I) = \{o \in \mathcal{O} \mid \forall i \in I \text{ nous avons } (o, i) \in \mathcal{R}\}$$

Le couple d'applications (ϕ, ψ) est une connexion de Galois entre l'ensemble des parties de $\mathcal{O}$ et l'ensemble des parties de $\mathcal{I}$. Étant donnée la connexion de Galois (ϕ, ψ), les propriétés suivantes sont vérifiées quelques soient les ensembles $I, I_1, I_2 \subseteq \mathcal{I}$ et $O, O_1, O_2 \subseteq \mathcal{O}$:

$$(1)\ I_1 \subseteq I_2 \Longrightarrow \psi(I_1) \supseteq \psi(I_2) \qquad (1')\ O_1 \subseteq O_2 \Longrightarrow \phi(O_1) \supseteq \phi(O_2)$$
$$(2)\ O \subseteq \psi(I) \Longleftrightarrow I \subseteq \phi(O)$$

[2] L'ensemble des parties d'un ensemble d'éléments $\mathcal{O}$, constitué de tous les sous-ensembles de $\mathcal{O}$, est noté $2^{\mathcal{O}}$.

[3] Un objet o contient un itemset l si o est en relation avec tous les items $i \in l$.

Exemple 4.1 Dans le contexte d'extraction de règles d'association $\mathcal{D}$ représenté dans la table 1.1, nous avons $\phi(\{1, 2\}) = \{C\}$ et $\psi(\{A, C\}) = \{1, 3, 5\}$ par exemple.

4.2.3 Propriétés des itemsets

Le support des itemsets et les propriétés 2.1 et 2.2 ont été définis de manière intuitive par Agrawal et al. dans [AIS93a, AS94]. L'application ψ de la connexion de Galois, qui associe à chaque itemset l'ensemble d'objets contenant l'itemset, permet de les définir formellement. Le support d'un itemset est la proportion d'objets du contexte qui contiennent l'itemset. Il correspond donc au rapport entre la taille de l'image de l'itemset par l'application ψ et la taille de l'ensemble $\mathcal{O}$.

Définition 4.1 (Support d'un itemset)
Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ et un itemset $l \subseteq \mathcal{I}$. Le support de l'itemset l dans le contexte $\mathcal{B}$ est :

$$support(l) = \frac{|\psi(l)|}{|\mathcal{O}|}.$$

Les propriétés 2.1 et 2.2 sont utilisées par les algorithmes d'extraction des itemsets fréquents et les algorithmes d'extraction des itemsets fréquents maximaux. Nous rappelons dans la suite ces propriétés et nous démontrons leur correction en utilisant l'application ψ.

Propriété 2.1 *Tous les sous-ensembles d'un itemset fréquent sont fréquents.*

Preuve. Soient $l, l' \subseteq \mathcal{I}$ deux itemsets tels que l est un itemset fréquent $l \in F$ et l' est un sous-ensemble de l, $l' \subseteq l$. Selon la propriété (1) de la connexion de Galois, nous avons $l' \subseteq l \Longrightarrow \psi(l') \supseteq \psi(l) \Longrightarrow support(l') \geq support(l) \geq minsupport$. Nous déduisons donc que l' est un itemset fréquent : $l' \in F$. □

Propriété 2.2 *Tous les sur-ensembles d'un itemset infréquent sont infréquents.*

Preuve. Soient $l, l' \subseteq \mathcal{I}$ deux itemsets tels que l est un itemset infréquent $l \notin F$ et l' est un sur-ensemble de l, $l' \supseteq l$. Selon la propriété (1) de la connexion de Galois, nous avons $l' \supseteq l \Longrightarrow \psi(l') \subseteq \psi(l) \Longrightarrow support(l') \leq support(l) < minsupport$. Nous déduisons donc que l' est un itemset infréquent : $l' \notin F$. □

4.2.4 Fermeture de la connexion de Galois

Nous considérons les ensembles des parties $2^{\mathcal{I}}$ et $2^{\mathcal{O}}$ dotés de la relation d'inclusion $\subseteq$, c'est à dire les ensembles partiellement ordonnés $(2^{\mathcal{I}}, \subseteq)$ et $(2^{\mathcal{O}}, \subseteq)$. Les opérateurs $\gamma = \phi \circ \psi$[4] de $(2^{\mathcal{I}}, \subseteq)$ dans $(2^{\mathcal{I}}, \subseteq)$ et $\gamma' = \psi \circ \phi$ de $(2^{\mathcal{O}}, \subseteq)$ dans $(2^{\mathcal{O}}, \subseteq)$ sont des *opérateurs de fermeture* de la connexion de Galois. Étant donnée la connexion de Galois (ϕ, ψ), les propriétés suivantes sont vérifiées quelques soient les ensembles $I, I_1, I_2 \subseteq \mathcal{I}$ et $O, O_1, O_2 \subseteq \mathcal{O}$:

(3) $I \subseteq \gamma(I)$	(3') $O \subseteq \gamma'(O)$
(4) $\gamma(\gamma(I)) = \gamma(I)$	(4') $\gamma'(\gamma'(O)) = \gamma'(O)$
(5) $I_1 \subseteq I_2 \Longrightarrow \gamma(I_1) \subseteq \gamma(I_2)$	(5') $O_1 \subseteq O_2 \Longrightarrow \gamma'(O_1) \subseteq \gamma'(O_2)$
(6) $\gamma'(\psi(I)) = \psi(I)$	(6') $\gamma(\phi(O)) = \phi(O)$

4.3 Treillis des itemsets fermés

Dans cette section, nous définissons les itemsets fermés, qui constituent le treillis des itemsets fermés, et les itemsets fermés fréquents.

4.3.1 Itemsets fermés

Étant donné l'opérateur de fermeture de la connexion de Galois γ, un itemset $l \subseteq \mathcal{I}$ tel que $\gamma(l) = l$ est appelé *itemset fermé*. Un itemset fermé est donc un ensemble maximal d'items communs à un ensemble d'objets.

Exemple 4.2 Par exemple, dans le contexte $\mathcal{D}$, l'itemset {BCE} est un itemset fermé car il est l'ensemble maximal d'items communs aux objets $\{2, 3, 5, 6\}$. L'itemset {BC} n'est pas un itemset fermé car il n'est pas un ensemble maximal d'items communs à certains objets : tous les objets contenant les items B and C (objets 2, 3,5 et 6) contiennent également l'item E. Dans une base de données de ventes, cela signifie que les clients achètent *au plus* les articles B, C et E et que tous les clients qui achètent les articles B et C achètent également l'article E.

[4]Nous utilisons les notations suivantes pour la composition d'applications : $\phi \circ \psi(I) = \phi(\psi(I))$ et $\psi \circ \phi(O) = \psi(\phi(O))$.

Définition 4.2 (Ensemble d'itemsets fermés)
Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ *et l'opérateur de fermeture de la connexion de Galois* γ. *L'ensemble* Γ *des itemsets fermés dans le contexte* $\mathcal{B}$ *est :*

$$\Gamma = \{l \subseteq \mathcal{I} \mid l \neq \varnothing \ \wedge \ \gamma(l) = l\}.$$

Remarque 4.1 Le plus petit (minimal au sens de l'inclusion) itemset fermé contenant un itemset l est obtenu par l'application de la fonction γ à l.

4.3.2 Treillis des itemsets fermés

Soit Γ l'ensemble des itemsets fermés dans un contexte $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ et la relation d'ordre partiel $\subseteq$ sur les éléments de Γ. Pour chaque couple d'éléments $l_1, l_2 \in \Gamma$ nous avons $l_1 \leq l_2$ si et seulement si $l_1 \subseteq l_2$. Nous appelons alors l_1 *sous-ensemble fermé* de l_2 et l_2 *sur-ensemble fermé* de l_1. L'ensemble partiellement ordonné $(\Gamma, \subseteq)$ forme un treillis complet car pour tout sous-ensemble $S \subseteq \Gamma$ il existe un plus petit majorant $Join(S)$ et un plus grand minorant $Meet$(S) définis comme suit [Wil82, Wil92] :

$$Join(S) = \gamma(\bigcup_{l \in S} l)$$
$$Meet(S) = \bigcap_{l \in S} l$$

Définition 4.3 (Treillis des itemsets fermés)
Soit Γ *l'ensemble des itemsets fermés du contexte* $\mathcal{B}$ *selon l'opérateur de fermeture de la connexion de Galois* γ. *Le couple* $\mathcal{L}_\Gamma = (\Gamma, \subseteq)$ *est un treillis complet appelé treillis des itemsets fermés.*

Exemple 4.3 Le treillis des itemsets fermés associé au contexte $\mathcal{D}$ est représenté dans la figure 4.1. Ce treillis contient 8 itemsets et sa hauteur est égale à cinq.

Remarque 4.2 Le treillis des itemsets fermés d'une relation binaire finie (le contexte d'extraction) est isomorphe au treillis formé par les composantes intensions des concepts du treillis de concept [Wil82, Wil92], également appelé treillis de Galois [GMA95], de cette relation binaire.

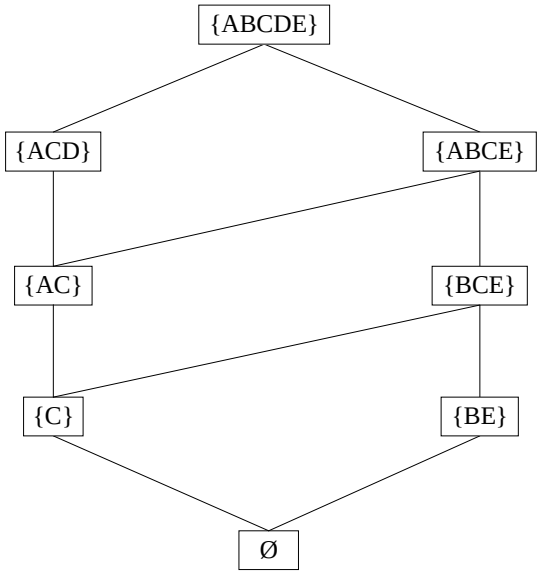


FIG. 4.1 – Treillis des itemsets fermés associé au contexte $\mathcal{D}$.

4.3.3 Itemsets fermés fréquents

Étant donné un seuil minimal de support *minsupport*, un itemset fermé $l \subseteq \mathcal{I}$ tel que support$(l) \geq$ *minsupport* est appelé *itemset fermé fréquent*.

Exemple 4.4 L'ensemble FF des itemsets fermés fréquents dans le contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 est présenté dans la table 4.1. L'itemset {BCE} est un itemset fermé fréquent pour $minsupport = 2/6$ car $support(\{BCE\}) = |\{2,3,5,6\}| / |\mathcal{O}| = 4/6 \geq minsupport$. Dans une base de données de ventes, cela signifie que 66% des clients achètent *au plus* les articles B, C et E.

Itemset fermé fréquent	Support
{C}	3/6
{BE}	5/6
{AC}	5/6
{BCE}	4/6
{ABCE}	2/6

TAB. 4.1 – Itemsets fermés fréquents extraits du contexte $\mathcal{D}$ pour $minsupport = 2/6$.

Définition 4.4 (Ensemble des itemsets fermés fréquents)
Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$. *Étant donné un seuil minimal de support* minsupport, *l'ensemble* FF *des itemsets fermés fréquents dans* $\mathcal{B}$ *est :*

$$FF = \{l \subseteq \mathcal{I} \mid l \neq \varnothing \ \wedge \ \gamma(l) = l \ \wedge \ \mathit{support}(l) \geq \text{minsupport}\}.$$

Un itemset fréquent est maximal si tous ses sur-ensembles sont infréquents. Dualement, un itemset fermé fréquent est maximal si tous ses sur-ensembles fermés sont infréquents.

Définition 4.5 (Ensemble des itemsets fermés fréquents maximaux)
Soit un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$. *Étant donné un seuil minimal de support* minsupport, *l'ensemble* FM *des itemsets fermés fréquents maximaux dans* $\mathcal{B}$ *est :*

$$FM = \{l \subseteq \mathcal{I} \mid l \in FF \ \wedge \ \forall l' \supset l \textit{ nous avons } \text{support}(l') < \text{minsupport}\}.$$

Exemple 4.5 Les itemsets fermés fréquents dans le contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 sont représentés dans la figure 4.2. Cinq itemsets sont fréquents parmi les sept itemsets fermés, l'itemset trivial $\varnothing$ n'étant pas considéré, et le plus grand des itemsets fermés fréquents est l'itemset fermé fréquent maximal {ABCE} dont la taille est quatre.

4.4 Approche par extraction des itemsets fermés fréquents

Nous proposons une nouvelle approche pour l'extraction de règles d'association basée sur l'extraction des itemsets fermés fréquents. Cette approche nous permet d'une part de réduire les temps d'extraction des itemsets fréquents, et d'autre part de générer seulement les règles d'association non redondantes. Elle est basée sur le théorème 4.1 concernant l'ensemble des itemsets fermés fréquents qui est un ensemble générateur pour les itemsets fréquents et leur support.

4.4.1 Base pour l'ensemble des itemsets fréquents

Afin de démontrer le théorème spécifiant que l'ensemble des itemsets fermés fréquents est un ensemble générateur non redondant, ou base, pour les itemsets

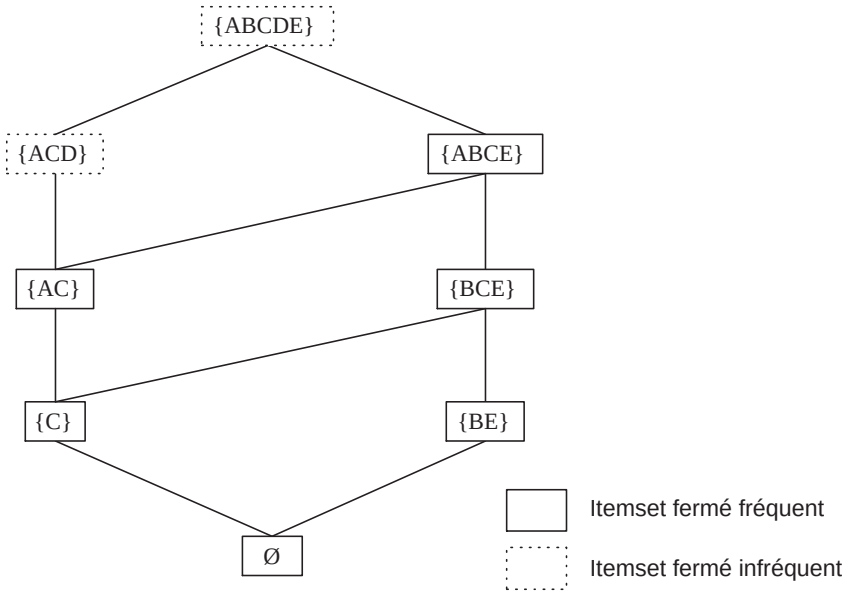


FIG. 4.2 – Itemsets fréquents dans le treillis des itemsets fermés associé au contexte $\mathcal{D}$ pour $minsupport = 2/6$.

fréquents et leurs supports, nous introduisons les quatre lemmes suivants concernant les itemsets fermés.

Lemme 4.1 *Tous les sous-ensembles d'un itemset fermé fréquent sont fréquents.*

Preuve. Cette propriété est directement dérivée de la propriété 2.1. □

Lemme 4.2 *Tous les sur-ensembles d'un itemset fermé infréquent sont infréquents.*

Preuve. Cette propriété est directement dérivée de la propriété 2.2. □

Lemme 4.3 *Le support d'un itemset l est égal au support de sa fermeture $\gamma(l)$ qui est le plus petit itemset fermé contenant l :* $\text{support}(l) = \text{support}(\gamma(l))$.

Preuve. Soit $l \subseteq \mathcal{I}$ un itemset et $\gamma(l)$ sa fermeture selon la connexion de Galois. Le support de l dans le contexte $\mathcal{B}$ est : $support(l) = |\psi(l)| \,/\, |\mathcal{O}|$. Puisque $\psi(l)$ est un itemset fermé et selon la propriété (6) de la connexion de Galois, nous avons $\gamma'(\psi(l)) = \psi(l)$. Nous déduisons :

$$support(\gamma(l)) = \frac{|\psi(\gamma(l))|}{|\mathcal{O}|} = \frac{|\psi(\phi(\psi(l)))|}{|\mathcal{O}|} = \frac{|\gamma'(\psi(l))|}{|\mathcal{O}|} = \frac{|\psi(l)|}{|\mathcal{O}|} = support(l). \quad \square$$

Lemme 4.4 *L'ensemble des itemsets fermés fréquents maximaux est identique à l'ensemble des itemsets fréquents maximaux : $M = FM$.*

Preuve. Il suffit de démontrer que tout itemset fréquent maximal $l \in M$ est un itemset fermé, c'est à dire que $\gamma(l) = l$. Selon la propriété (3) de la connexion de Galois, la fermeture d'un itemset fréquent maximal $l \in M$ est un sur-ensemble de cet itemset : $l \subseteq \gamma(l)$. D'après le lemme 4.3, nous avons $support(\gamma(l)) = support(l)$ et puisque l est fréquent, $\gamma(l)$ est également fréquent : $support(\gamma(l)) = support(l) \geq minsupport$. Or, puisque l est maximal, nous en concluons que $l = \gamma(l) : l$ est un itemset fermé fréquent maximal. Tout itemset fréquent maximal étant un itemset fermé, les ensembles M et FM sont identiques. □

Théorème 4.1 (Base pour les itemsets fréquents)
L'ensemble des itemsets fermés fréquents est une base pour l'ensemble des itemsets fréquents.

Preuve. Selon les lemmes 4.2 et 4.1, tous les sous-ensembles d'un itemset fermé fréquent sont fréquents et tous les sur-ensembles d'un itemset fermé infréquent sont infréquents. L'ensemble des itemsets fermés fréquents maximaux, qui est identique à l'ensemble des itemsets fréquents maximaux selon le lemme 4.4, forme donc une bordure au-dessous de laquelle tous les itemsets sont fréquents et au-dessus de laquelle tous les itemsets sont infréquents. Il est donc possible d'énumérer tous les itemsets fréquents en générant tous les sous-ensembles des itemsets fermés fréquents. De plus, selon le lemme 4.3, le support d'un itemset qui n'est pas fermé peut être déduit du support de sa fermeture qui est le plus petit itemset fermé le contenant. Ainsi, les itemsets fréquents et leurs supports peuvent être générés à partir des itemsets fermés fréquents et leurs supports, sans accéder au contexte d'extraction. □

Corollaire 4.1 (Base pour les règles d'association valides)
L'ensemble des itemsets fermés fréquents est une base pour l'ensemble des règles d'association valides.

Preuve. L'ensemble des itemsets fréquents est une base pour les règles d'association valides, car ces dernières, leurs supports et leurs confiances sont générées à partir des itemsets fréquents et de leurs supports sans accéder au contexte d'extraction.

Puisque l'ensemble des itemsets fermés fréquents est une base pour l'ensemble des itemsets fréquents, il est également une base pour les règles d'association valides : ces dernières peuvent être générées à partir des itemsets fermés fréquents et leurs supports, et leurs confiances peuvent être déterminés à partir des supports des itemsets fermés fréquents. □

Remarque 4.3 L'ensemble des itemsets fréquents maximaux, qui est égal à l'ensemble des itemsets fermés fréquents maximaux, n'est pas une base pour l'ensemble des itemsets fréquents. En effet, s'il est possible d'énumérer tous les itemsets fréquents en générant tous les sous-ensembles des itemsets fréquents maximaux, le support des itemsets fréquents ne peut pas être déduit de celui des itemsets fréquents maximaux. En conséquence, l'ensemble des itemsets fréquents maximaux n'est pas une base pour l'ensemble des règles d'association valides car le support et la confiance des règles d'association ne peuvent être déduits des itemsets fréquents maximaux.

4.4.2 Nouvelle approche pour l'extraction des itemsets fréquents

Soient un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ et un seuil minimal de support *minsupport* défini par l'utilisateur. Étant donné le théorème 4.1, le problème de l'extraction des itemsets fréquents dans le contexte $\mathcal{B}$ peut être décomposé en deux sous-problèmes :

1. Déterminer l'ensemble des itemsets fermés fréquents dans $\mathcal{B}$, c'est à dire les itemsets fermés dont le support est supérieur ou égal à *minsupport*.
2. Dériver l'ensemble des itemsets fréquents à partir de l'ensemble des itemsets fermés fréquents extrait durant la phase 1. Cette phase consiste à générer tous les sous-ensembles des itemsets fermés fréquents maximaux et dériver leurs supports à partir des supports des itemsets fermés fréquents.

Seul le premier sous-problème nécessite la réalisation de balayages du contexte d'extraction. Utilisant cette nouvelle décomposition du problème, il est possible d'améliorer considérablement les temps de réponse de l'extraction des itemsets fréquents, car nous limitons l'espace de recherche des itemsets fréquents au treillis des itemsets

fermés, qui est un sous-ordre du treillis des itemsets. En effet, la proportion d'itemsets qui sont des itemsets fermés fréquents est dans la plupart des cas réels très inférieure à la proportion d'itemsets fréquents, et dans ce cas la hauteur du treillis des itemsets fermés et bien inférieure à celle du treillis des itemsets fréquents. En minimisant ainsi la taille de l'espace de recherche, nous réduisons le nombre de balayages et le nombre d'opérations internes réalisées lors de l'extraction des itemsets fréquents.

Si le problème de l'extraction des itemsets fermés fréquents a une complexité exponentielle dans la taille de l'ensemble des items dans le pire des cas, des résultats théoriques et expérimentaux ont démontrés que la complexité en moyenne est bien inférieure dans le cas de bases de données réelles [GM94, GMA95]. Comme nous l'avons vu précédemment, la taille du treillis des itemsets $\mathcal{L}_{\mathcal{I}}$ est exponentielle dans la taille de l'ensemble des items : $|\mathcal{L}_{\mathcal{I}}| = 2^{|\mathcal{I}|}$. Dans le pire des cas, c'est à dire si tous les itemsets sont des itemsets fermés, la taille du treillis des itemsets fermés est égale à la taille du treillis des itemsets : $|\mathcal{L}_{\Gamma}| \leq 2^{|\mathcal{I}|}$. Toutefois, la taille du treillis des itemsets fermés est linéaire dans la taille du contexte d'extraction $\mathcal{B}$ s'il existe une borne supérieure K sur la taille $|\phi(\{o\})|$ des objets. Dans ce cas, la taille de ce treillis est majorée par : $|\mathcal{L}_{\Gamma}| \leq 2^K|\mathcal{B}|$ [GM94, GMA95]. Cette condition est vérifiée dans la plupart des bases de données réelles. De plus, dans l'hypothèse d'une distribution uniforme, des résultats expérimentaux et théoriques ont démontré que la taille de ce treillis peut être majorée par : $|\mathcal{L}_{\Gamma}| \leq \mu|\mathcal{B}|$, μ étant la valeur moyenne du nombre d'items par objets $Moy(|\phi(\{o\})|)$ [GMA95]. Les gains en termes de temps d'exécution seront donc encore plus importants si la distribution des données est uniforme, condition qui est vérifiée dans de nombreux jeux de données tels que les jeux de données statistiques par exemple.

4.4.3 Nouvelle approche pour l'extraction de règles d'association

Soient un contexte d'extraction $\mathcal{B} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ et deux seuils minimaux de support *minsupport* et de confiance *minconfiance* définis par l'utilisateur. Étant donné le corollaire 4.1, le problème de l'extraction des règles d'association valides dans le contexte $\mathcal{B}$ peut être décomposé en deux sous-problèmes :

1. Déterminer l'ensemble des itemsets fermés fréquents dans $\mathcal{B}$, c'est à dire les

itemsets fermés dont le support est supérieur ou égal à *minsupport*.

2. Générer les règles d'association valides dont la confiance est supérieure ou égale à *minconfiance* à partir des itemsets fermés fréquents extraits lors de la phase 1.

Cette décomposition permet de réduire les temps d'extraction des règles d'association en utilisant les itemsets fermés fréquents, au lieu des itemsets fréquents, pour générer les règles valides. Ici également, seul le premier sous-problème nécessite la réalisation de balayages du contexte d'extraction, les règles étant ensuite générées sans accéder au contexte.

L'utilisation de l'ensemble des itemsets fermés fréquents permet également d'extraire des *couvertures réduites*, également appelées *bases*, pour les règles d'association. Ces bases sont des ensembles de tailles réduites qui minimisent le nombre de règles d'association générées tout en maximisant la quantité et la qualité des informations convoyées par les règles. Les itemsets fermés fréquents constituent l'élément central de ces bases qui permettent d'améliorer la pertinence du résultat de l'extraction car elles ne contiennent aucunes règles d'association redondantes. Elles permettent également de présenter à l'utilisateur un ensemble réduit de règles couvrant l'ensemble des items du contexte et donc de découvrir des règles « inattendues » qui constituent des informations utiles qu'il est nécessaire de considérer. La génération de bases pour les règles d'association en utilisant les itemsets fermés fréquents est présentée dans le chapitre 6.

4.5 Discussion

Plusieurs travaux concernant la définition de cadres formels pour l'extraction des règles d'association ont été réalisés. Dans [GKMT97, MT97], le problème est formalisé dans la théorie des langages afin d'analyser la complexité de ce problème et d'établir des connexions avec les problèmes de l'extraction de séries chronologiques, des contraintes d'intégrité dans les bases de données relationnelles et de la recherche des transversaux minimaux d'un hypergraphe. Dans [FGLS98], un cadre formel basé sur le langage logique constitué de l'ensemble des clauses de Horn sur un alphabet donné est défini et les connexions avec la logique probabiliste du premier ordre de Halpern, la programmation logique inductive et la formalisation du problème dans

la théorie des langages sont étudiées. Un cadre formel pour l'extraction des règles d'association basé sur *l'analyse formelle de concepts* introduite par Wille [Wil82], a été proposé concurremment à nos travaux dans [ZO98]. Toutefois, ce cadre utilise le treillis de concepts, dans lequel chaque concept possède une composante extension qui est un ensemble d'objets et une composante intension qui est un ensemble d'items, et aucune propriété permettant de réduire le treillis de concepts au treillis des itemsets fermés fréquents (composante intension seulement pour les concepts fréquents seulement) n'est définie. De plus, aucune propriété permettant de développer un algorithme efficace d'extraction des concepts dans le cadre du KDD, c'est à dire pour de grands jeux de données, n'est proposée. De même, aucune propriété et aucun algorithme efficace de génération des règles d'association valides ou bien de bases pour les règles d'association valides à partir du treillis de concepts ne sont proposés.

A partir de la sémantique pour le problème de l'extraction de règles d'association que nous proposons, nous déduisons deux nouvelles approches pour l'extraction des itemsets fréquents et l'extraction de bases pour les règles d'association. Ces approches sont basées sur l'extraction des itemsets fermés, selon la fermeture de la connexion de Galois, qui sont fréquents dans le contexte. Nous utilisons également cette sémantique pour en déduire des propriétés permettant de définir des algorithme efficaces d'extraction des ensembles fermés fréquents dans le cadre des contraintes du KDD et de génération de bases pour les règles d'association. Ces propriétés ainsi que ces algorithmes sont présentés dans les chapitres 5 et 6. Les propriétés que nous avons définies, pour le problème de l'extraction de règles d'association peuvent également être utilisées pour d'autres problèmes du KDD. En effet, plusieurs travaux ont démontrés que le problème de l'extraction des itemsets fréquents est un sous-problème commun à plusieurs autres problèmes du KDD tels que la découverte de séries chronologiques, le clustering et la classification. Les aspects algorithmiques de l'utilisation de l'extraction des itemsets fermés fréquents pour ces problèmes sont discutés plus en détail dans la section 5.4. De plus amples travaux sont nécessaires afin de proposer une extension de la sémantique que nous proposons pour formaliser ces problèmes et en déduire éventuellement des propriétés permettant d'améliorer la résolution de ces problèmes.

Chapitre 5

Découverte des ensembles fermés fréquents d'items

Sommaire

5.1 Introduction

La découverte des itemsets fermés fréquents consiste à extraire du treillis des itemsets fermés, selon la fermeture de la connexion de Galois γ, associé à $\mathcal{I}$ les itemsets dont le support dans le contexte d'extraction est supérieur ou égal au seuil minimal de support *minsupport*. Nous proposons deux nouveaux algorithmes d'extraction des itemsets fermés fréquents nommés Close [PBTL98, PBTL99c] et

A-Close [PBTL99a]. Ces algorithmes réalisent un parcours en largeur du treillis des itemsets fermés pour en extraire les itemsets fermés fréquents et leurs supports. Ils utilisent en entrée le contexte d'extraction sur lequel sont réalisés des balayages et le seuil minimal de support *minsupport*. Nous proposons également un troisième algorithme, nommé Close$^+$ [PBTL99a], qui identifie les itemsets fermés fréquents et leurs supports à partir des itemsets fréquents et leurs supports déjà extraits du contexte. Cet algorithme utilise en entrée la collection des ensembles d'itemsets fréquents, chaque ensemble contenant les itemsets fréquents d'un taille donnée.

Comme les algorithmes d'extraction des itemsets fréquents et des itemsets fréquents maximaux, les algorithmes Close, A-Close et Close$^+$ utilisent un ordre total sur les itemsets. Cet ordre est l'ordre lexicographique, introduit pour les algorithmes Apriori et OCD, présenté dans la section 2.1. Il permet de limiter le nombre d'opérations réalisées lors de l'extraction, en parcourant chaque itemset fermé du treillis au plus une seule fois. De plus, il permet de construire des structures de données autorisant l'exécution efficace des opérations portant sur les itemsets.

5.2 Algorithmes d'extraction des itemsets fermés fréquents

Les algorithmes Close et A-Close, présentés dans les sections 5.2.1 et 5.2.2 respectivement, sont des algorithmes itératifs d'extraction des itemsets fermés fréquents. Ces algorithmes sont basés sur la notion d'*itemsets générateurs* des itemsets fermés. Ces itemsets générateurs sont utilisés afin de construire un ensemble d'itemsets fermés candidats (itemsets fermés potentiellement fréquents) qui sont les fermetures des générateurs.

Définition 5.1 (Itemsets générateurs d'un itemset fermé)
Un itemset générateur g d'un itemset fermé $f = \gamma(f)$ est un itemset minimal (selon la relation d'inclusion) dont la fermeture par l'opérateur γ est l'itemset fermé $\gamma(g) = f$. L'ensemble G_f des générateurs d'un itemset fermé f est :

$$G_f = \{g \subseteq \mathcal{I} \mid \gamma(g) = f \ \wedge \ \nexists g' \subset g \text{ tel que } \gamma(g') = f\}.$$

Exemple 5.1 Dans le contexte $\mathcal{D}$, l'ensemble d'itemsets générateurs de l'itemset fermé {BE} est $G_{\{\text{BE}\}} = \{\{\text{B}\}, \{\text{E}\}\}$ car $\gamma(\{\text{B}\}) = \gamma(\{\text{E}\}) = \{\text{BE}\}$. Pour l'itemset

fermé {BCE}, nous avons $G_{\{BCE\}} = \{\{BC\}, \{CE\}\}$ car aucun 1-itemset n'a pour fermeture {BCE} et la fermeture de l'itemset {BE} est $\gamma(\{BE\}) = \{BE\}$.

Remarque 5.1 Soit $g \subseteq \mathcal{I}$ un itemset générateur d'un itemset fermé $f = \gamma(f)$. Selon le lemme 4.3, le support de g est égal au support de sa fermeture : $support(g) = support(f)$.

Durant chaque itération, l'algorithme Close considère un ensemble de k-itemsets générateurs. Il construit un ensemble d'itemsets fermés candidats qui sont les fermetures de ces k-générateurs[1] et il détermine ensuite parmi ces candidats les itemsets fermés fréquents selon le seuil minimal de support *minsupport*. Finalement, il crée les $(k+1)$-générateurs qui seront utilisés lors de l'itération suivante afin de construire l'ensemble d'itemsets fermés candidats qui sont les fermetures des $(k+1)$-générateurs. Un balayage du contexte d'extraction est nécessaire durant chaque itération, afin de déterminer les fermetures des k-générateurs et calculer leurs supports.

L'algorithme A-Close commence par déterminer les 1-itemsets générateurs fréquents. Ensuite, durant chaque itération k, il génère un ensemble de k-générateurs candidats à partir des $(k$-1)-itemsets générateurs fréquents. Il détermine les supports de ces k-itemsets générateurs candidats et supprime les générateurs infréquents et les générateurs non minimaux qui sont identifiés en fonction de leurs supports. Lorsque tous les générateurs fréquents sont déterminés, leurs fermetures qui constituent les itemsets fermés fréquents sont déterminées. Durant chaque itération, un balayage du contexte est réalisé afin de calculer le support des k-générateurs candidats. Un ultime balayage du contexte est ensuite réalisé lors de la détermination des fermetures de tous les générateurs fréquents.

L'algorithme Close$^+$, présenté dans la section 5.2.3, permet de déterminer les itemsets fermés fréquents parmi l'ensemble des itemsets fréquents, sans accéder au contexte d'extraction. Il permet d'étendre une implémentation actuelle d'extraction des itemsets fréquent, sans avoir à la modifier, afin de générer les itemsets fermés fréquents. Les k-itemsets fréquents qui sont des k-itemsets fermés fréquents sont identifiés en comparant leurs supports avec les supports de leurs $(k+1)$-sur-ensembles. Cet algorithme ne réalise pas de balayage car il s'applique à l'ensemble des itemsets fréquents et leurs supports déjà extrait.

[1] Nous appelons k-générateur un itemset générateur de taille k.

5.2.1 Close

L'algorithme Close [PBTL98, PBTL99c], proposé en 1998, est un algorithme d'extraction des itemsets fermés fréquents qui parcours l'ensemble des générateurs des itemsets fermés fréquents par niveaux. L'ensemble d'itemsets fermés candidats d'une itération k est l'ensemble des fermetures des k-générateurs de cette itération. L'ensemble de 1-générateurs est initialisé avec la liste des 1-itemsets du contexte durant la première itération. Ensuite, pour chaque itération k correspondant au $k^{ème}$ niveau :

1. la fermeture de chaque k-générateur et le support du générateur et de sa fermeture sont calculés ;
2. pour chaque k-générateur fréquent, sa fermeture (qui est un itemset fermé fréquent), et leur support sont insérés dans l'ensemble des itemsets fermés fréquents ;
3. un ensemble de $(k+1)$-générateurs (utilisés dans l'itération suivante) est construit en utilisant les k-générateurs fréquents de l'ensemble des itemsets fermés fréquents.

Durant chaque itération, un balayage du contexte d'extraction est réalisé afin de déterminer les fermetures des générateurs et les supports des générateurs et de leurs fermetures. Le pseudo-code de l'algorithme Close est présenté dans l'algorithme 5.1. Les notations utilisées sont présentées dans la table 5.1.

FFC_k	Ensemble de k-groupes candidats des k-générateurs. Chaque élément de cet ensemble possède trois champs : *générateur*, *fermé* et *support*.
FF_k	Ensemble de k-groupes fréquents des k-générateurs. Chaque élément de cet ensemble possède trois champs : *générateur*, *fermé* et *support*.

TAB. 5.1 – Notations utilisées dans l'algorithme Close.

Chaque élément c de l'ensemble FFC_k est un *k-groupe candidat* contenant un k-générateur, la fermeture de ce générateur et leur support dans le contexte. Les fermetures des générateurs de FFC_k constituent les itemsets fermés fréquents candidats de l'itération k. Chaque élément e de l'ensemble FF_k est un *k-groupe fréquent* contenant un k-générateur fréquent, la fermeture (fréquente) de ce générateur et leur support. Nous notons $FFC_k.$*générateurs* l'ensemble des itemsets générateurs contenus dans

les champs *générateur* de FFC_k. Les notations $FFC_k.fermés$ et $FFC_k.supports$ sont définies de manière identique.

ALG. 5.1 Extraction des itemsets fermés fréquents avec Close.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ;
Sortie : ensembles FF_k des k-groupes fréquents ;

1) $FFC_1.générateurs \leftarrow \{1\text{-}itemsets\}$;
2) **pour** ($k \leftarrow 1$; $FFC_k.générateurs \neq \varnothing$; k++) **faire**
3) Gen-Closure(FFC_k) ;
4) **pour chaque** groupe candidat $c \in FFC_k$ **faire**
5) **si** ($c.support \geq minsupport$) **alors** $FF_k \leftarrow FF_k \cup \{c\}$;
6) **fin pour**
7) $FFC_{k+1} \leftarrow$ Gen-Generator(FF_k) ;
8) **fin pour**
9) **retourner** $\bigcup_k FF_k$;

Durant la première itération de l'algorithme (ligne 1), l'ensemble de 1-générateurs de FFC_1 est initialisé avec les 1-itemsets du contexte, c'est à dire les éléments de l'ensemble $\mathcal{I}$. Cette initialisation ne nécessite aucun balayage du contexte. Chacune des itérations k suivantes (lignes 2 à 8) consiste en trois phases. La première phase consiste à déterminer les fermetures des k-générateurs dans l'ensemble FFC_k et calculer le support de chacun des itemsets fermés fréquents candidats ainsi obtenus (ligne 3). Cette phase est réalisée par la procédure Gen-Closure. Durant la seconde phase, les itemsets fermés fréquents candidats dont le support est supérieur ou égal au seuil minimal de support *minsupport* sont insérés dans l'ensemble FF_k des itemsets fermés fréquents de la $k^{ème}$ itération (lignes 4 à 6). Durant le troisième phase, les $(k+1)$-générateurs de l'ensemble FFC_{k+1} sont créés en appliquant de la procédure Gen-Generator aux k-générateurs de l'ensemble FF_k (ligne 7). Ces itérations sont répétées jusqu'à ce que l'ensemble FFC_{k+1} généré soit vide, c'est à dire que aucun nouvel itemset générateur ne peut être généré. Tous les itemsets fermés fréquents dans le contexte ont alors été générés et le résultat de l'algorithme Close est correct (voir Théorème 5.1).

Procédure Gen-Closure(FFC_k) La procédure Gen-Closure reçoit un ensemble FFC_k de k-groupes candidats contenant les k-générateurs candidats de l'itération k en argument. Elle détermine la fermeture de chaque générateur, stockée dans le champ *fermé*, et leur support stocké dans le champ *support*. Le pseudo-code de la procédure est présenté dans l'algorithme 5.2. La détermination des fermetures des générateurs est basée sur la proposition 5.1.

Proposition 5.1 *L'itemset fermé $\gamma(l)$ correspondant à la fermeture selon l'opérateur γ de l'itemset l est l'intersection de tous les objets du contexte contenant l :*

$$\gamma(l) = \bigcap_{o \in \mathcal{O}} \{\phi(\{o\}) \mid l \subseteq \phi(\{o\})\}.$$

Preuve. Soit $H = \bigcap_{o \in S} \phi(\{o\})$ avec $S = \{o \in O \mid l \subseteq \phi(\{o\})\}$. Nous avons $\gamma(l) = \phi(\psi(l)) = \bigcap_{o \in \psi(l)} \phi(\{o\}) = \bigcap_{o \in S'} \phi(\{o\})$ avec $S' = \{o \in O \mid o \in \psi(l)\}$. Il suffit de démontrer que $S' = S$:

$$\begin{aligned} l \subseteq \phi(\{o\}) &\iff o \in \psi(l) \\ o \in \psi(l) &\iff l \subseteq \phi(\psi(l)) \subseteq \phi(\{o\}). \end{aligned}$$

Nous en concluons que $S = S'$ et donc que $\gamma(l) = H$. □

Selon la proposition 5.1, un seul balayage du contexte est nécessaire afin de déterminer les fermetures de tous les générateurs d'une itération et leurs supports. La procédure Gen-Generator considère successivement chaque objet du contexte (lignes 3 à 10). Pour chaque objet o, l'ensemble G_o est créé (ligne 4) en utilisant la fonction Subset décrite dans la section 2.2.1. G_o contient tous les itemsets générateurs de FFC_k qui sont des sous-ensembles de l'itemset $\phi(\{o\})$. Ensuite, pour chaque générateur *g.générateur* dans G_o, sa fermeture *g.fermé* et leur support *g.support* sont mis à jour comme suit (lignes 5 à 9). Si l'objet o considéré est le premier objet parcouru contenant *g.générateur* alors *g.fermé* est vide et est initialisé avec l'itemset $\phi(\{o\})$ (ligne 6). Sinon, l'intersection entre l'itemset *g.fermé* et l'itemset $\phi(\{o\})$ donne la nouvelle valeur de *g.fermé* (ligne 7). Dans les deux cas, le support *g.support* des itemsets *g.générateur* et *g.fermé* est incrémenté (ligne 8). Lorsque tous les objets du contexte ont été considérés, la procédure retourne l'ensemble FFC_k mis à jour avec pour chaque générateur, sa fermeture et leur support. Le résultat de la procédure Gen-Closure est correct car l'itemset *g.fermé* associé à chaque générateur *g.générateur* est l'intersection de tous les objets contenant *g.générateur*.

Le support *g.support* associé aux itemsets *g.générateur* et *g.fermé* correspond au nombre d'objets contenant *g.fermé* et donc contenant *g.générateur* car *g.générateur* $\subseteq \gamma(g.générateur) = g.fermé$.

ALG. 5.2 Calcul des fermetures et des supports des générateurs avec Gen-Closure.

Entrée : ensembles FFC_k des k-groupes candidats ; contexte $\mathcal{B}$;
Sortie : champs *fermé* et *support* des candidats de FFC_k mis à jour ;

1) $FFC_k.fermés \leftarrow \varnothing$;
2) $FFC_k.supports \leftarrow 0$;
3) **pour chaque** objet $o \in \mathcal{B}$ **faire**
4) $\quad G_o \leftarrow \text{Subset}(FFC_k.générateurs, \phi(\{o\}))$;
5) $\quad$ **pour chaque** générateur $g.générateur \in G_o$ **faire**
6) $\quad\quad$ **si** $(g.fermé = \varnothing)$ **alors** $g.fermé \leftarrow \phi(\{o\})$;
7) $\quad\quad$ **sinon** $g.fermé \leftarrow g.fermé \cap \phi(\{o\})$;
8) $\quad\quad$ $g.support++$;
9) $\quad$ **fin pour**
10) **fin pour**
11) **retourner** $\bigcup\{g \in FFC_k \mid g.fermé \neq \varnothing\}$;

Procédure Gen-Generator(FF_k) La procédure Gen-Generator reçoit un ensemble FF_k de k-groupes fréquents en paramètre. Elle retourne l'ensemble FFC_{k+1} des $(k+1)$-groupes candidats contenant les $(k+1)$-générateurs qui seront utilisés durant l'itération $k+1$. Le pseudo-code de la procédure est présenté dans l'algorithme 5.3. La création des générateurs est basée sur le lemme 5.1.

La procédure Gen-Generator est constituée de trois phases. Durant la première phase, tous les $(k+1)$-générateurs potentiels sont créés en utilisant les k-générateurs dans FF_k. Les secondes et troisièmes phases permettent ensuite de supprimer parmi ces générateurs ceux dont on sait que le calcul de la fermeture est inutile. La seconde phase supprime les générateurs potentiels infréquents et ceux qui ne sont pas minimaux (qui ne sont donc pas des générateurs). La troisième phase se base sur le lemme 5.3 pour supprimer parmi ces générateurs ceux dont la fermeture à déjà été calculée.

Lemme 5.1 *Quelques soient deux itemsets $l_1, l_2 \subseteq \mathcal{I}$ quelconques, nous avons :*

$$\gamma(l_1 \cup l_2) = \gamma(\gamma(l_1) \cup \gamma(l_2)).$$

Preuve. Soient l_1 et l_2 deux itemsets. Selon la propriété (3) de la fermeture de la connexion de Galois, nous avons :

$$l_1 \subseteq \gamma(l_1) \text{ et } l_2 \subseteq \gamma(l_2) \Longrightarrow l_1 \cup l_2 \subseteq \gamma(l_1) \cup \gamma(l_2) \Longrightarrow \gamma(l_1 \cup l_2) \subseteq \gamma(\gamma(l_1) \cup \gamma(l_2)) \quad (a)$$

Puisque $l_1 \subseteq l_1 \cup l_2$ et $l_2 \subseteq l_1 \cup l_2$ (évident), nous déduisons que $\gamma(l_1) \subseteq \gamma(l_1 \cup l_2)$ et $\gamma(l_2) \subseteq \gamma(l_1 \cup l_2)$. Selon la propriété (4) de la fermeture de la connexion de Galois, nous avons :

$$\gamma(\gamma(l_1) \cup \gamma(l_2)) \subseteq \gamma(\gamma(l_1 \cup l_2)) \Longrightarrow \gamma(\gamma(l_1) \cup \gamma(l_2)) \subseteq \gamma(l_1 \cup l_2) \quad (b)$$

Des résultats (a) et (b) nous concluons que $\gamma(l_1 \cup l_2) = \gamma(\gamma(l_1) \cup \gamma(l_2))$. □

Lemme 5.2 *Soit un itemset $l_1 \subseteq \mathcal{I}$ et $l_2 \subset l_1$ un sous-ensemble de l_1 tel que $l_1 \subseteq \gamma(l_2)$. Nous avons alors $\gamma(l_1) = \gamma(l_2)$ et $\forall l_3 \subseteq \mathcal{I}$, $\gamma(l_1 \cup l_3) = \gamma(l_2 \cup l_3)$.*

Preuve. Soient $l_1, l_2 \subseteq \mathcal{I}$ deux itemsets tels que $l_2 \subset l_1 \subseteq \gamma(l_2)$. Selon les propriétés d'isotonie (4) et d'idempotence (5) de l'opérateur de fermeture γ nous déduisons que $\gamma(l_2) \subseteq \gamma(l_1) \subseteq \gamma(\gamma(l_2)) = \gamma(l_2)$ et nous en concluons que $\gamma(l_1) = \gamma(l_2)$. Pour un itemset $l_3 \subseteq \mathcal{I}$, selon le lemme 5.1 nous avons : $\gamma(l_1 \cup l_3) = \gamma(\gamma(l_1) \cup \gamma(l_3)) = \gamma(\gamma(l_2) \cup \gamma(l_3)) = \gamma(l_2 \cup l_3)$. □

Lemme 5.3 *Soit un k-itemset l et l'ensemble $\mathcal{S} = \{s_1, s_2, \ldots, s_j\}$ des (k-1)-sous-ensembles de l tels que $\bigcup_{s \in \mathcal{S}} s = l$. S'il existe un sous-ensemble $s_a \in \mathcal{S}$ tel que $l \subseteq \gamma(s_a)$, alors $\gamma(l) = \gamma(s_a)$. l n'est pas un générateur (non minimal) de $\gamma(l)$.*

Preuve. Dérivée du lemme 5.2. □

Durant la première phase de la procédure Gen-Generator (lignes 1 à 4), l'ensemble FFC_{k+1} est initialisé en appliquant la phase de jointure de la procédure Apriori-Gen [AS94] aux générateurs fréquents de FF_k : Deux k-générateurs fréquents p et q de FF_k sont joints si les $k-1$ premiers items qui les composent sont identiques. Le résultat de cette union est un $(k+1)$-générateur potentiel qui est inséré dans FFC_{k+1}. Les générateurs potentiels inutiles sont ensuite supprimés de FFC_{k+1} selon deux stratégies.

La seconde phase consiste, comme dans la procédure Apriori-Gen, à vérifier pour chaque générateur potentiel créé la présence de tous ses-ensembles de taille k dans FF_k (lignes 5 à 9). Cette phase permet de supprimer deux types de générateurs : ceux qui sont des sur-ensemble de générateurs infréquents et sont donc infréquents eux-mêmes ; ceux qui sont inclus dans la fermeture d'un de leurs sous-ensemble qui a donc déjà été générée. Supposons par exemple que l'ensemble FF_2 reçu en paramètre par la procédure contienne les générateurs fréquents {AB} et {AC}. La première phase de la procédure créera le générateur potentiel {ABC} = {AB} $\cup$ {AC} dans l'ensemble FFC_3 et ce dernier sera supprimé car {BC} n'est pas présent dans FF_2.

La troisième phase consiste à tester pour chaque générateur potentiel g s'il est inclus dans la fermeture d'un des k-générateurs s de l'ensemble FF_k. Si tel est le cas, la fermeture de g est égale à la fermeture de s selon le lemme 5.3 et il peut être supprimé de FFC_{k+1} (lignes 10 à 15). Supposons par exemple que l'ensemble FF_2 reçu en paramètre par la procédure contienne les générateurs fréquents {FG}, {FH} et {GH} et leurs fermetures fréquentes respectives {FG}, {FGHI} et {GH}. Le générateur potentiel {FGH} = {FG} $\cup$ {FH} sera créé lors de la première phase de la procédure dans l'ensemble FFC_3. La troisième phase supprimera le générateur {FGH} de FFC_3 car {FGH} $\subseteq \gamma(\{FH\}) = \{FGHI\}$ et donc $\gamma(\{FGH\}) = \gamma(\{FG\})$. Le 3-générateur {FGH} est donc inutile et supprimé de FFC_3.

Procédure Subset(G, l) Les ensembles de groupes candidats et groupes fréquents sont stockés dans un *prefix-tree* [Bas00, Mue95] afin d'accélérer les recherches des générateurs contenus dans un objet. La figure 5.1 présente la structure du prefix-tree associé à un ensemble FFC_2 de 2-groupes candidats contenant les 2-générateurs AB et BC. Chaque arc de l'arbre est nommé par un item. Un générateur est représenté par un chemin dans l'arbre en partant du noeud racine. La fermeture d'un générateur est stockée dans le noeud feuille qui termine le chemin représentant le générateur. Chaque noeud contient un pointeur sur un noeud frère, une table de hachage vers ses noeuds fils et, si le noeud est une feuille, un pointeur vers la fermeture du générateur représenté. Pour un noeud représentant un k-générateur g, un noeud frère représente un autre k-générateur qui possède les même premiers $k-1$ items et un collision de hachage sur le $k^{ème}$ item. Pour des raisons de performances, si la taille d'une liste chaînée de noeuds frères dépasse une valeur seuil, la taille de la table de hachage du

ALG. 5.3 Création des générateurs avec Gen-Generator.

Entrée : ensemble FF_k de k-groupes fréquents ;

Sortie : ensemble FFC_{k+1} avec les champs *générateur* des $(k+1)$-groupes candidats initialisés ;

// Phase 1
1) **insert into** $FFC_{k+1}.générateur$
2) **select** $p[1],\ p[2],\ \ldots,\ p[k],\ q[k]$
3) **from** $FF_k.générateurs\ p,\ FF_k.générateurs\ q$
4) **where** $p[1] = q[1],\ \ldots,\ p[k-1] = q[k-1],\ p[k] < q[k]$;
// Phase 2
5) **pour chaque** générateur $g.générateur \in FFC_{k+1}$ **faire**
6) **pour chaque** k-sous-ensemble s of $g.générateur$ **faire**
7) **si** $(s \notin FF_k.générateurs)$ **alors supprimer** g de FFC_{k+1} ;
8) **fin pour**
9) **fin pour**
// Phase 3
10) **pour chaque** générateur $g.générateur \in FFC_{k+1}$ **faire**
11) $S_g \leftarrow \text{Subset}(FF_k.générateurs, g.générateur)$;
12) **pour chaque** $s \in S_g$ **faire**
13) **si** $(g.générateur \subseteq s.fermé)$ **alors supprimer** g de FFC_{k+1} ;
14) **fin pour**
15) **fin pour**
16) Retourner FFC_{k+1} ;

noeud parent est doublée et les $k^{èmes}$ noeuds sont ré-équilibrés.

La procédure Subset reçoit un ensemble de générateurs G et un itemset l comme paramètres. Elle détermine quels générateurs $g \in G$ sont des sous-ensembles de l'itemset l. La recherche des sous-ensembles est effectuée en débutant par le noeud racine du prefix-tree et réalisant un parcours en profondeur du prefix-tree en appliquant la fonction de hachage à chaque item de l'itemset l successivement. Lorsque un noeud a été atteint en « hachant » un item i de l, la fonction de hachage est appliquée aux items venant après i dans l, et ce processus est appliqué récursivement à l'élément de la table de hachage ainsi atteint. Pour le noeud racine du prefix-

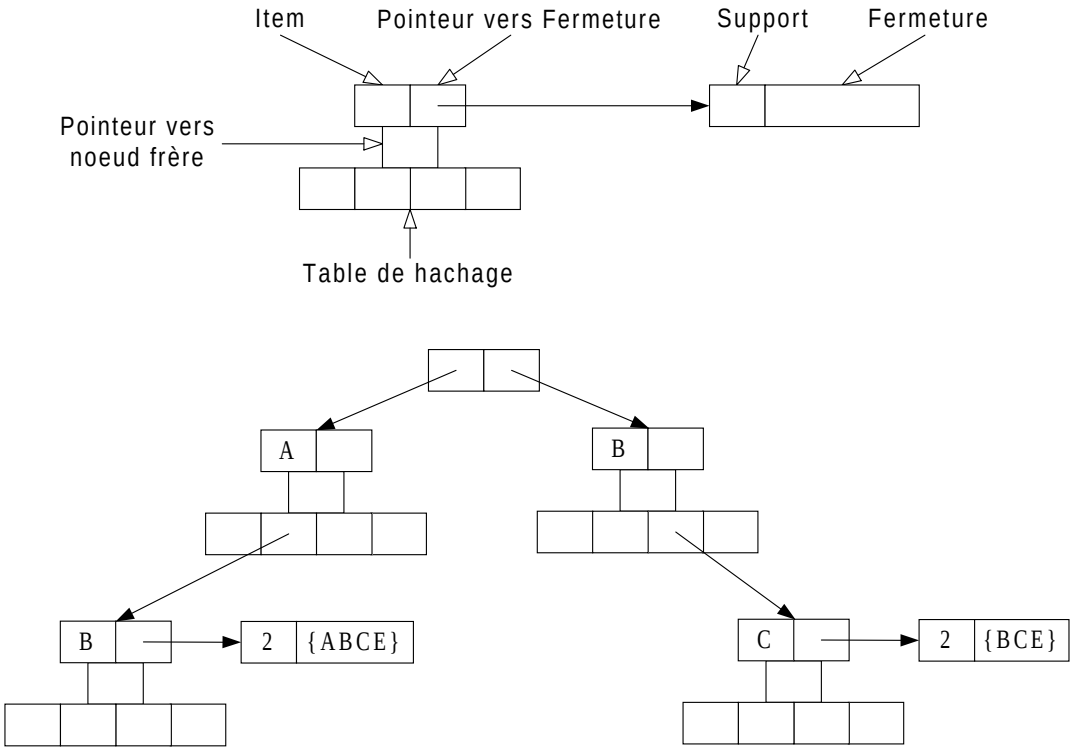


FIG. 5.1 – Prefix-tree associé à un ensemble FFC_2 de 2-groupes candidats.

tree, la fonction de hachage est appliquée à tous les items de l. Lorsqu'un noeud feuille du prefix-tree est atteint, le générateur représenté par le chemin parcourus est un sous-ensemble de l et une référence vers ce générateur est ajoutée à l'ensemble résultat.

Exemple 5.2 La figure 5.2 représente l'exécution de l'algorithme Close au contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6. L'ensemble FFC_1 est initialisé avec la liste des 1-itemsets du contexte $\mathcal{D}$ (ligne 1). La procédure Gen-Closure génère les fermetures des 1-générateurs, qui sont les itemsets fermés fréquents potentiels, et leurs supports dans FFC_1 (ligne 5). Les groupes candidats de FFC_1 qui sont fréquents sont insérés dans l'ensemble FF_1 (lignes 6 à 8). La première phase de la procédure Gen-Generator (ligne 9) à l'ensemble FF_1 génère six nouveaux 2-générateurs candidats : {AB}, {AC}, {AE}, {BC}, {BE} et {CE} dans FFC_2. Les 2-générateurs {AC} et {BE} sont supprimés de FFC_2 par la seconde phase de la procédure Gen-Generator car nous avons {AC} $\subseteq \gamma(\{A\})$ et {BE} $\subseteq$

Balayage de $\mathcal{D}$ $\longrightarrow$

FFC_1

Générateur	Fermé	Support
{A}	{AC}	3/6
{B}	{BE}	5/6
{C}	{C}	5/6
{D}	{ACD}	1/6
{E}	{BE}	5/6

Suppression des itemsets infréquents $\longrightarrow$

FF_1

Générateur	Fermé	Support
{A}	{AC}	3/6
{B}	{BE}	5/6
{C}	{C}	5/6
{E}	{BE}	5/6

Balayage de $\mathcal{D}$ $\longrightarrow$

FFC_2

Générateur	Fermé	Support
{AB}	{ABCE}	2/6
{AE}	{ABCE}	2/6
{BC}	{BCE}	4/6
{CE}	{BCE}	4/6

Suppression des itemsets infréquents $\longrightarrow$

FF_2

Générateur	Fermé	Support
{AB}	{ABCE}	2/6
{AE}	{ABCE}	2/6
{BC}	{BCE}	4/6
{CE}	{BCE}	4/6

FIG. 5.2 – Extraction des itemsets fermés fréquents dans le contexte $\mathcal{D}$ avec Close pour $minsupport = 2/6$.

$\gamma(\{B\})$. La procédure Gen-Closure calcule ensuite les fermetures et les supports des 2-générateurs restant dans FFC_2 et les ensembles FF_2 et FFC_2 sont identiques car tous les itemsets fermés de FFC_2 sont fréquents. L'application de la procédure Gen-Generator à l'ensemble FF_2 génère le 3-générateur {ABE} qui est supprimé car le 2-générateur {BE} n'appartient pas à FF_2 et l'algorithme s'arrête. On peut observer que le nombre de balayages du contexte réalisés par l'algorithme Close est inférieur de moitié au nombre de balayages réalisés par l'algorithme Apriori pour cet exemple. De plus, l'algorithme Close détermine le support de 8 itemsets seulement alors que l'algorithme Apriori calcule le support de 16 itemsets, diminuant ainsi le nombre de tests d'inclusion des itemsets dans les objets nécessaires.

Remarque 5.2 Il est possible d'optimiser l'algorithme en ne conservant que le premier 1-générateur dans l'ordre lexicographique parmi ceux qui possèdent une fermeture identique. Les 1-générateurs extraits du contexte $\mathcal{D}$ pour $minsupport = 2/6$ sont alors {A}, {B} et {C}. Le générateur {E} qui possède une fermeture identique à l'itemset {B} n'est pas conservé et seulement trois 2-générateurs candidats sont créés dans FFC_2 : {AB}, {AC} et {BC}. Ensuite, seuls les générateurs fréquents

{AB} et {BC} sont insérés dans FF_2 et aucun 3-générateur candidat n'est créé dans FFC_3.

Correction de l'algorithme Dans ce paragraphe, nous démontrons la correction de l'algorithme Close énoncée dans le théorème 5.1. Nous introduisons auparavant le lemme 5.4 qui sera utilisé afin d'en établir la preuve.

Lemme 5.4 *Pour chaque itemset $g \subseteq \mathcal{I}$ tel que $|g| > 1$, si g n'est pas dans l'ensemble $FF_{|g|}$ (des $|g|$-générateurs fréquents) et si g est un itemset fréquent alors il existe deux sous-ensembles s_1 et s_2 de g de taille $|s_1| = |s_2| - 1$ tels que $\gamma(s_1) = \gamma(s_2)$ et s_1 est un générateur dans l'ensemble $FF_{|s_1|}$.*

Preuve. Nous utilisons une démonstration par récurrence. Pour $|g| = 2$, nous avons $s_2 = g$ et $\exists s_1 \in FF_1$ tel que $s_1 \subset s_2$ et $\gamma(s_1) = \gamma(s_2)$ (le lemme 5.4 est évident). Supposons que le lemme 5.4 est vérifié pour $|g| = k$, nous démontrons qu'il est alors vérifié pour $|g| = k + 1$. Soit le générateur g de taille $|g| = k + 1$ tel que $g \notin FF_{|g|}$. Deux cas sont alors possibles :
(a) $\exists g' \subset g$ tel que $|g'| = k$ et $g' \notin FF_{|g'|}$.
(b) $\exists g' \subset g$ tel que $|g'| = k$ et $g' \in FF_{|g'|}$ et $g \subseteq \gamma(g') \Rightarrow \gamma(g) = \gamma(g')$ (lemme 5.2).
Si (a) est vérifié alors selon l'hypothèse de récurrence $\exists s_1 \subset s_2 \subseteq g' \subset g$ tel que $\gamma(s_1) = \gamma(s_2)$ et $s_1 \in FF_{|s_1|}$. Si (b) est vérifié alors nous identifions s_1 à g' et s_2 à g. □

Théorème 5.1 (L'algorithme Close est correct)
L'algorithme Close génère tous les itemsets fermés fréquents et leurs supports.

Preuve. Nous démontrons par récurrence que pour tout itemset $l \subseteq \mathcal{I}$ fréquent, la fermeture de l est dans l'ensemble résultat FF : $\gamma(l) \in FF$. L'hypothèse de récurrence est vérifiée pour les 1-itemsets qui correspondent aux 1-générateurs g insérés dans FFC_1 et dont la fermeture $\gamma(g)$ est insérée dans FF_1 si $support(g) \geq minsupport$, et donc $\gamma(g) \in FF$. Supposons maintenant que $\forall l \subseteq \mathcal{I}$ tel que $|l| = k$ nous avons $\gamma(l) \in FF$. Nous démontrons alors que $\forall l \subseteq \mathcal{I}$ de taille $|l| = k + 1$ nous avons $\gamma(l) \in FF$. Soit un itemset l de taille $|l| = k + 1$. Deux cas sont alors possibles :
(a) Si $l \in FF_{|l|}$ alors $\gamma(l) \in FF_{|l|}$ (évident) et donc $\gamma(l) \in FF$.
(b) Si $l \notin FF_{|l|}$ alors selon le lemme 5.4, nous avons : $\exists s_1 \subset s_2 \subseteq l$ tel que $s_1 \in FF_{|s_1|}$

et $\gamma(s_1) = \gamma(s_2)$. Puisque $\gamma(l) = \gamma(s_2 \cup l \setminus s_2) = \gamma(s_1 \cup l \setminus s_2)$ et $|s_1 \cup l \setminus s_2| = k$ alors, conformément à l'hypothèse de récurrence, nous concluons que $\gamma(s_1 \cup l \setminus s_2) \in FF_k$ et donc $\gamma(l) \in FF$. $\square$

5.2.2 A-Close

L'algorithme A-Close [PBTL99b], proposé en 1999, est un algorithme d'extraction des itemsets fermés fréquents utilisant les propriétés des supports des générateurs des itemsets fermés fréquents. Il génère itérativement les k-générateurs des k-groupes fréquents des ensembles FF_k pour k variant de 1 à μ comme suit. L'ensemble de 1-générateurs fréquents est initialisé avec les 1-itemsets fréquents du contexte et ensuite, durant une itération k :

1. un ensemble de (k+1)-générateurs candidats est créé à partir des k-générateurs fréquents ;
2. le support de tous les (k+1)-générateurs candidats est déterminé ;
3. les (k+1)-générateurs candidats g dont le support est égal au support d'un k-générateur qui est un sous-ensemble de g sont supprimés.

Durant chaque itération, un balayage du contexte est réalisé afin de calculer le support des (k+1)-générateurs candidats. Lorsque tous les générateurs fréquents sont déterminés, la fermeture de chacun d'eux est calculée en réalisant un balayage du contexte. Le pseudo-code est présenté dans l'algorithme 5.4 et les notations utilisées sont présentées dans la table 5.2.

FF_k	Ensemble de k-groupes fréquents des k-générateurs. Chaque élément de cet ensemble possède trois champs : *générateur*, *fermé* et *support*.

TAB. 5.2 – Notations utilisées dans l'algorithme A-Close.

Durant la première itération de l'algorithme (ligne 1), l'ensemble des 1-générateurs de FF_1 est initialisé avec la liste des 1-itemsets du contexte. La procédure Support-Count est ensuite appliquée afin de déterminer les supports de ces 1-itemsets générateurs en réalisant un balayage du contexte (ligne 2) et les 1-générateurs infréquents sont supprimés de FF_1 (lignes 3 à 5). Durant chaque itération k suivante (lignes 6 à 8), les (k+1)-générateurs de l'ensemble FF_{k+1} sont créés en utilisant les

ALG. 5.4 Extraction des itemsets fermés fréquents avec A-Close.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ;

Sortie : ensembles FF_k des k-groupes fréquents ;

1) $FF_1.générateurs \leftarrow \{1\text{-}itemsets\}$;
2) Support-Count($\mathcal{B}, FF_1.générateurs$) ;
3) **pour chaque** générateur $ggénérateur \in FF_1$ **faire**
4) **si** ($g.support < minsupport$) **alors supprimer** g de FF_k ;
5) **fin pour**
6) **pour** ($k \leftarrow 1$; $FF_k.générateurs \neq \varnothing$; $k{+}{+}$) **faire**
7) $FF_{k+1} \leftarrow$ AC-Generator(FF_k) ;
8) **fin pour**
9) AC-Closure($\bigcup FF_k$) ;
10) **retourner** $\bigcup_k FF_k$;

k-générateurs de l'ensemble FF_k et de leurs supports respectifs. La procédure AC-Generator est pour cela appliquée à l'ensemble FF_k (ligne 7). Ces itérations cessent lorsque aucun nouvel itemset générateur ne peut être créé. Tout les générateurs des ensembles FF_k ont alors été créés et la procédure AC-Closure est appliquée à l'ensemble de ces générateurs afin de déterminer leurs fermetures qui constituent les itemsets fermés fréquents (ligne 9). L'algorithme retourne finalement la collection des ensembles FF_k qui contiennent chacun tous les k-générateurs et leurs fermetures (ligne 10).

Procédure AC-Generator(FF_k) La procédure AC-Generator reçoit un ensemble FF_k de k-groupes fréquents contenant les k-générateurs fréquents en paramètre. Elle retourne l'ensemble FF_{k+1} de (k+1)-groupes fréquents contenant les (k+1)-générateurs fréquents. Le pseudo-code de la procédure est présenté dans l'algorithme 5.5.

La procédure AC-Generator est constituée de trois phases dont les deux premières sont identiques aux deux premières phases de la procédure Gen-Generator. Durant la première phase, les k-générateurs fréquents de FF_k sont combinés afin de créer les (k+1)-générateurs potentiels dans FF_{k+1}. La seconde phase supprime les (k+1)-générateurs potentiels infréquents ou qui ne sont pas minimaux. La troisième phase

est basée sur le lemme 5.5 afin de supprimer les (k+1)-générateurs potentiels restant dont un sous-ensemble est un générateur du même itemset fermé fréquent.

Lemme 5.5 *Soit un k-générateur g et l'ensemble $\mathcal{S} = \{s_1, s_2, \ldots, s_j\}$ des (k-1)-sous-ensembles de g tel que $\bigcup_{s \in \mathcal{S}} s = g$. Si il existe un sous-ensemble $s_a \in \mathcal{S}$ tel que $support(s_a) = support(g)$, alors $\gamma(g) = \gamma(s_a)$.*

Preuve. Dérivée du lemme 5.2. □

La première phase de la procédure (lignes 1 à 4) applique la phase de jointure de la procédure Apriori-Gen aux k-générateurs de FF_k afin d'initialiser les (k+1)-générateurs potentiels de FF_{k+1}. La seconde phase (lignes 5 à 9) vérifie la présence dans FF_k de tous les k-générateurs qui sont des sous-ensembles de chaque (k+1)-générateur potentiels dans FF_{k+1}. Durant la troisième phase, un balayage du contexte est réalisé afin de déterminer le support de chaque (k+1)-générateur potentiels restant dans FF_{k+1} (ligne 10) et tous les (k+1)-générateurs de FF_{k+1} sont examinés (lignes 11 à 18). Si un (k+1)-générateurs g est infréquent, il est supprimé de FF_{k+1} (ligne 12). Sinon, si il existe un générateur s de l'ensemble FF_k qui est un sous-ensemble de g et qui possède le même support que g, alors g est supprimé de FF_{k+1} (ligne 15). En effet, selon le lemme 5.5, g et s possèdent la même fermeture et g n'est pas un itemset minimal dont la fermeture est $\gamma(s) = \gamma(g)$. Supposons par exemple que FF_2 contienne les 2-générateurs fréquents {AB}, {AC} et {BC} de supports respectifs 3/6, 2/6 et 3/6. Un 3-générateur potentiel {ABC} sera créé dans FF_3 et supposons que son support soit égal à 2. Lors de la troisième phase, {ABC} sera supprimé de FF_3 car $support(\{ABC\}) = support(\{AC\})$ et donc $\gamma(\{ABC\}) = \gamma(\{AC\})$.

Procédure AC-Closure(FF) La procédure AC-Closure reçoit un ensemble $FF = \bigcup FF_k$ de groupes fréquents contenant tous les générateurs fréquents en argument. Elle détermine la fermeture de chaque générateur dans le champ *fermé* du groupe fréquent en réalisant un balayage du contexte. La méthode utilisée est identique à celle de la procédure Gen-Closure qui est basée sur la proposition 5.1. Le pseudo-code de la procédure est présenté dans l'algorithme 5.6.

La procédure AC-Closure traite chaque objet du contexte successivement (lignes 1 à 7) et créée pour chaque objet o un ensemble G_o (ligne 2) contenant tous les générateurs de FF qui sont des sous-ensembles de l'itemset $\phi(\{o\})$. Ensuite, pour chaque

ALG. 5.5 Création des générateurs avec AC-Generator.

Entrée : ensemble FF_k de k-groupes des k-générateurs fréquents ;

Sortie : ensemble FFC_{k+1} avec les champs *générateur* des $(k+1)$-groupes candidats initialisés ;

```
    // Phase 1
 1) insert into FF_{k+1}.générateur
 2) select p[1], p[2], ..., p[k], q[k]
 3) from FF_k.générateurs p, FF_k.générateurs q
 4) where p[1] = q[1], ..., p[k-1] = q[k-1], p[k] < q[k] ;
    // Phase 2
 5) pour chaque générateur g.générateur ∈ FFC_{k+1} faire
 6)       pour chaque k-sous-ensemble s de g.générateur faire
 7)             si (s ∉ FF_k.générateurs) alors supprimer g de FFC_{k+1} ;
 8)       fin pour
 9) fin pour
    // Phase 3
10) Support-Count(B, FF_1.générateurs) ;
11) pour chaque générateur g.générateur ∈ FFC_{k+1} faire
12)       si (g.support < minsupport) alors supprimer g de FF_{k+1} ;
13)       sinon faire
14)             pour chaque k-sous-ensemble s.générateur ∈ FFC_k de g faire
15)                   si (s.support = g.support) alors supprimer g de FFC_{k+1} ;
16)             fin pour
17)       fin pour
18) fin pour
19) Retourner FFC_{k+1} ;
```

générateur *g.générateur* dans G_o, la fermeture *g.fermé* est mise à jour (lignes 3 à 6). Lorsque tous les objets du contexte ont été considérés, la procédure retourne l'ensemble FF_k dans lequel les champs *fermé* qui sont les fermetures de générateurs fréquents sont mis à jour.

Exemple 5.3 L'exécution de l'algorithme A-Close sur le contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6 est représentée dans la figure 5.3. L'ensemble

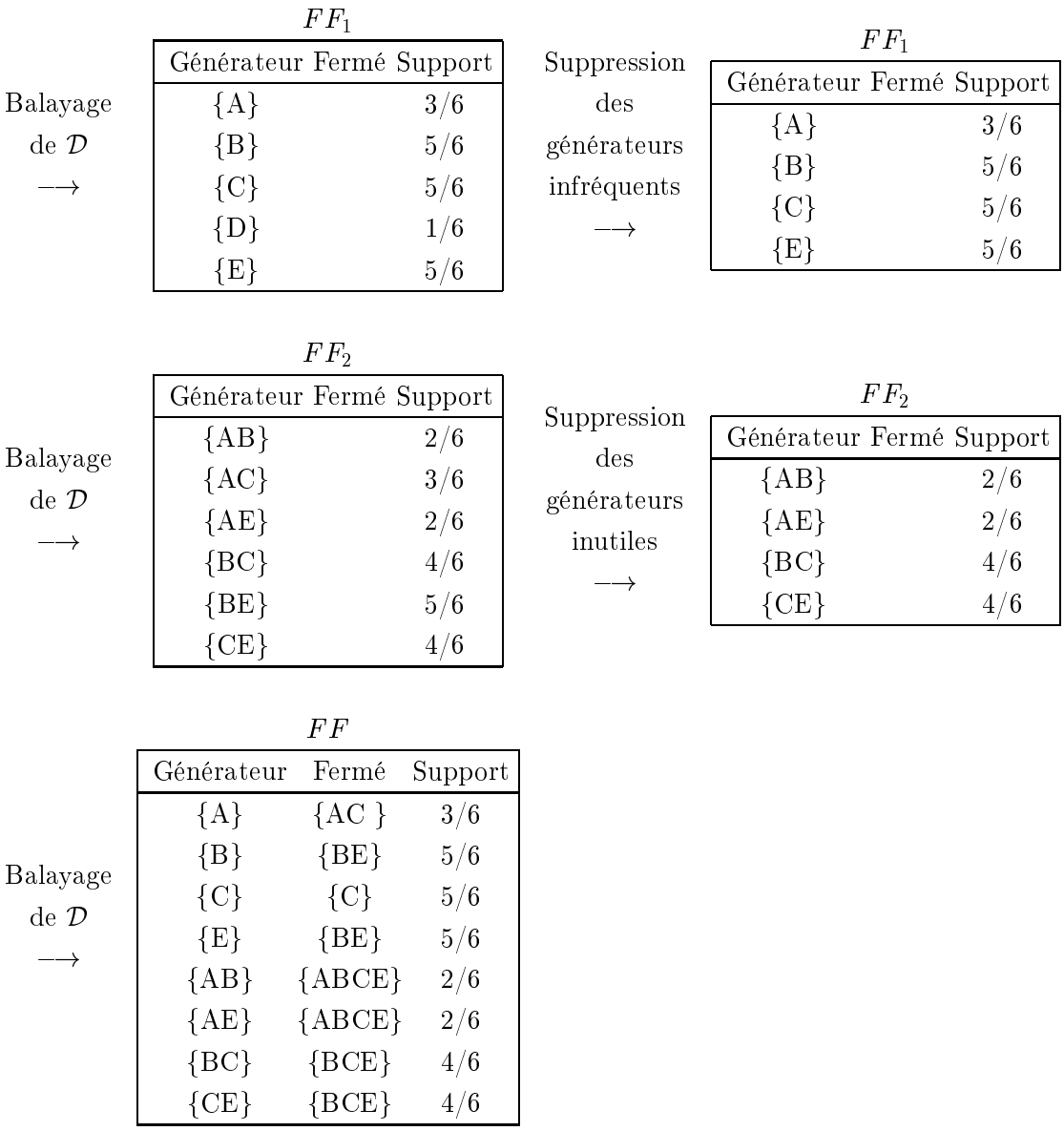
ALG. 5.6 Calcul des fermetures des générateurs avec AC-Closure.

Entrée : ensembles FF_k des k-groupes des k-générateurs fréquents ; contexte $\mathcal{B}$;
Sortie : champs *fermé* des groupes de FF_k mis à jour ;

1) **pour chaque** objet $o \in \mathcal{B}$ **faire**
2) $G_o \leftarrow \text{Subset}(FF_k.\textit{générateurs}, \phi(\{o\}))$;
3) **pour chaque** générateur $g.\textit{générateur} \in G_o$ **faire**
4) **si** $(g.\textit{fermé} = \varnothing)$ **alors** $g.\textit{fermé} \leftarrow \phi(\{o\})$;
5) **sinon** $g.\textit{fermé} \leftarrow g.\textit{fermé} \cap \phi(\{o\})$;
6) **fin pour**
7) **fin pour**
9) **retourner** $\bigcup\{g \in FF_k\}$;

FF_1 est initialisé avec la liste des 1-itemsets du contexte $\mathcal{D}$ (ligne 1) et la procédure Support-Count détermine le support de chacun d'eux en réalisant un balayage du contexte (ligne 2). Les groupes candidats de FF_1 qui sont infréquents sont supprimés de l'ensemble FF_1 (lignes 3 à 5). L'application de la procédure AC-Generator aux générateurs de l'ensemble FF_1 génère six 2-générateurs potentiels qui sont insérés dans FF_2 (ligne 7). Le 2-générateurs potentiel {AC} est supprimé de FF_2 car son support est égal au support du 1-générateur {A} et donc $\gamma(\{AC\}) = \gamma(\{A\})$. De même, le 2-générateurs potentiel {BE} qui possède un support identique aux 1-générateurs {B} et {E} est supprimé de FF_2. L'application de la procédure AC-Generator à l'ensemble FF_2 génère le 3-générateur potentiel {ABE} qui est supprimé avant le balayage du contexte car {BE} n'est pas un générateur de FF_2 et les itérations cessent. Un ultime balayage du contexte est réalisé par la procédure AC-Closure (ligne 9) afin de calculer les fermetures des générateurs créés dans FF_1 et FF_2 qui sont les itemsets fermés fréquents du contexte.

Remarque 5.3 Il est possible d'optimiser l'algorithme en mémorisant le numéro de la première itération durant laquelle un générateur fréquent qui n'est pas un itemset fermé fréquent est identifié. Le numéro de cette itération correspond à la taille t de ce générateur non-fermé et il n'est pas nécessaire de déterminer la fermeture des générateurs de tailles inférieures puisque tous sont des itemsets fermés fréquents : ils sont eux-même leur propres générateurs uniques. L'ensemble de générateurs passé en paramètre à la procédure AC-Closure est alors réduit à l'union des ensembles de



FF_1

Générateur	Fermé	Support
{A}		3/6
{B}		5/6
{C}		5/6
{D}		1/6
{E}		5/6

FF_1

Générateur	Fermé	Support
{A}		3/6
{B}		5/6
{C}		5/6
{E}		5/6

FF_2

Générateur	Fermé	Support
{AB}		2/6
{AC}		3/6
{AE}		2/6
{BC}		4/6
{BE}		5/6
{CE}		4/6

FF_2

Générateur	Fermé	Support
{AB}		2/6
{AE}		2/6
{BC}		4/6
{CE}		4/6

FF

Générateur	Fermé	Support
{A}	{AC }	3/6
{B}	{BE}	5/6
{C}	{C}	5/6
{E}	{BE}	5/6
{AB}	{ABCE}	2/6
{AE}	{ABCE}	2/6
{BC}	{BCE}	4/6
{CE}	{BCE}	4/6

FIG. 5.3 – Extraction des itemsets fermés fréquents dans le contexte $\mathcal{D}$ avec A-Close pour $minsupport = 2/6$.

générateurs de taille supérieure à t, ce qui permet de réduire le nombre d'opérations réalisées par la procédure. Cette optimisation est implémentée dans la version de l'algorithme A-Close utilisée pour les expérimentations.

Théorème 5.2 (L'algorithme A-Close est correct)
L'algorithme A-Close génère tous les itemsets fermés fréquents et leurs supports.

Preuve. L'algorithme A-Close fonctionne selon un procédé identique au procédé utilisé par l'algorithme Close. La différence entre les deux algorithmes provient de la condition de suppression des $(k+1)$-générateurs potentiels de FF_{k+1} utilisée dans la troisième phase des procédures Gen-Generator de Close et AC-Generator de A-close. Dans la procédure Gen-Generator, un $(k+1)$-générateur potentiel g inclus dans la fermeture d'un de ses k-sous-ensembles s qui est un générateur de FF_k est supprimé car nous avons alors $\gamma(g) = \gamma(s)$. Cette condition correspond au lemme 5.3. Dans la procédure AC-Generator, un $(k+1)$-générateur potentiel g dont le support est égal au support d'un de ses k-sous-ensembles s qui est un générateur de FF_k est supprimé car nous avons alors également $\gamma(g) = \gamma(s)$. Cette condition correspond au lemme 5.5. Ces deux conditions étant strictement équivalentes, les algorithmes Close et A-Close gênèrent les mêmes k-générateurs dans les ensembles FF_k et la correction de l'algorithme Close (théorème 5.1) assure la correction de l'algorithme A-close. □

5.2.3 Close$^+$

L'algorithme Close$^+$ proposé dans [PBTL99a], permet de déterminer les itemsets fermés fréquents en utilisant l'ensemble des itemsets fréquents et leur support, sans accéder au contexte d'extraction. Le pseudo-code est présenté dans l'algorithme 5.7 et les notations utilisées sont présentées dans la table 5.3. La méthode utilisée consiste à identifier les itemsets fréquents qui sont des itemsets fermés fréquents en comparant leurs supports avec les supports des itemsets fréquents qui sont leurs sur-ensembles. Cette méthode est directement dérivée de la propriété 5.1.

Propriété 5.1 *Le support d'un itemset fermé est supérieur au support de chacun de ses sur-ensembles stricts.*

Preuve. Soit un itemset fermé $f = \gamma(f)$ et un ensemble E contenant tous les itemsets dont la fermeture est l'itemset f : $E = \{l \subseteq \mathcal{I} \mid \gamma(l) = f\}$. Selon le lemme 4.3, tous les itemsets de E possèdent un support identique : $\forall l_1, l_2 \in E$, $support(l_1) = support(l_2)$. Soit un itemset s qui est un sur-ensemble strict de f, c'est à dire tel que $s \supset f$. Selon la propriété (1) de la connexion de Galois, $s \supset f \Longrightarrow \psi(s) \subset \psi(f)$.

Nous en déduisons que $|\psi(s)| < |\psi(f)| \Longrightarrow support(s) < support(f)$. Le support d'un sur-ensemble strict s de l'itemset fermé f est inférieur au support de f. □

Les (k-1)-itemsets fermés fréquents sont déterminés à partir des (k-1)-itemsets fréquents, des k-itemsets fréquents et de leurs supports. Les (k-1)-itemsets fréquents dont le support est différent des supports de tous leurs sur-ensembles parmi les k-itemsets fréquents sont des (k-1)-itemsets fermés fréquents et sont insérés dans l'ensemble des (k-1)-itemsets fermés fréquents. Le lemme 4.4 nous permet de déterminer directement les plus longs itemsets fermés fréquents de taille μ qui sont les μ-itemsets fréquents.

F_k	Ensemble des k-itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
Γ_k	Ensemble de k-itemsets fermés fréquents. Chaque élément de cet ensemble possède deux champs : *fermé* et *support*.
isclosed	Variable globale indiquant si l'itemset fréquent f considéré est fermé ou non.

TAB. 5.3 – Notations utilisées dans l'algorithme Close$^+$.

L'algorithme examine successivement les ensembles F_k de k-itemsets fréquents et F_{k-1} de (k-1)-itemsets fréquents en partant des ensembles F_μ et $F_{\mu-1}$ et jusqu'aux ensembles F_2 et F_1. Lors de chaque itération, les (k-1)-itemsets fréquents dans F_{k-1}, dont le support est différent du support de tous leurs sur-ensembles dans F_k, sont insérés dans l'ensemble Γ_{k-1} des (k-1)-itemsets fermés fréquents. Le lemme 4.4 permet de déterminer l'ensemble Γ_μ des plus plus longs itemsets fermés fréquents qui est identique à l'ensemble F_μ des plus longs itemsets fréquents.

L'algorithme commence par initialiser l'ensemble Γ_μ des μ-itemsets fermés fréquents avec l'ensemble F_μ des μ-itemsets fréquents (ligne 1). L'algorithme détermine ensuite itérativement quels (k-1)-itemsets fréquents de l'ensemble F_{k-1} ont un support différent des supports de leurs k-sur-ensembles fréquents (lignes 2 à 11). Au début de la $k^{ème}$ itération, l'ensemble Γ_{k-1} est initialisé avec l'ensemble vide (ligne 3). Ensuite, le support de chaque (k-1)-itemset fréquent f est comparé avec les supports de ses sur-ensembles dans F_k (lignes 4 à 10). Si le support de f est égal au support de au moins un de ses sur-ensembles, il n'est pas fermé et la variable *isclosed* prend la valeur "faux" (ligne 7). Sinon la variable *isclosed* possède la valeur "vrai" et f

ALG. 5.7 Identification des itemsets fréquents qui sont fermés avec Close$^+$.

Entrée : ensembles F_k des k-itemsets fréquents ;
Sortie : ensemble Γ_k des k-itemsets fermés fréquents ;

1) $\Gamma_\mu \leftarrow F_\mu$;
2) **pour** ($k \leftarrow \mu$; $k > 1$; k--) **faire**
3) $\quad \Gamma_{k-1} \leftarrow \varnothing$;
4) $\quad$ **pour chaque** (k-1)-itemset fréquent $f \in F_{k-1}$ **faire**
5) $\quad\quad$ *isclosed* $\leftarrow$ vrai ;
6) $\quad\quad$ **pour chaque** k-itemset fréquent $s \in F_k$ **tel que** ($f \subset s$) **faire**
7) $\quad\quad\quad$ **si** ($f.support = s.support$) **alors** *isclosed* $\leftarrow$ faux ;
8) $\quad\quad$ **fin pour**
9) $\quad\quad$ **si** (*isclosed* = vrai) **alors** $\Gamma_{k-1} \leftarrow \Gamma_{k-1} \cup \{f\}$;
10) $\quad$ **fin pour**
11) **fin pour**
12) **Retourner** $\bigcup_k \Gamma_k$;

est inséré dans l'ensemble Γ_{k-1} des (k-1)-itemsets fermés fréquents (ligne 9). Les itérations cessent lorsque tous les ensembles Γ_k pour $1 \leq k \leq \mu$ ont été générés.

Théorème 5.3 (L'algorithme Close$^+$ est correct)
L'algorithme Close$^+$ génère tous les itemsets fermés fréquents ainsi que leurs supports à partir des itemsets fréquents et leurs supports.

Preuve. La correction de la détermination des ensembles Γ_{k-1} des (k-1)-itemsets fermés fréquents pour $k \leq \mu$ découle de la propriété 5.1. Les (k-1)-itemsets fréquents qui ne sont pas insérés dans l'ensemble Γ_{k-1} sont ceux dont le support est égal au support d'un de leurs k-sur-ensembles et qui ne sont donc pas des itemsets fermés fréquents. La correction de la détermination de l'ensemble Γ_μ est assurée par la propriété 4.4. Les itemsets fréquents maximaux étant des itemsets fermés fréquents maximaux, l'ensemble F_μ des plus longs itemsets fréquents est identique à l'ensemble Γ_μ des plus longs itemsets fermés fréquents. En effet, ces itemsets sont par définition des itemsets maximaux : $F_\mu = \Gamma_\mu \subseteq M = FM$, pour M l'ensemble des itemsets fréquents maximaux et FM l'ensemble des itemsets fermés fréquents maximaux. □

Dualement, les générateurs fréquents peuvent être identifiés de manière directe parmi les itemsets fréquents en utilisant la propriété 5.2. Les k-itemsets générateurs fréquents sont alors déterminés à partir des k-itemsets fréquents, des (k-1)-itemsets fréquents et de leurs supports. Les k-itemsets fréquents dont le support est différent des supports de tous leurs sous-ensembles parmi les (k-1)-itemsets fréquents sont des k-générateurs fréquents.

Propriété 5.2 *Le support d'un itemset générateur est inférieur au support de chacun de ses sous-ensembles stricts*[2].

Preuve. Soit un itemset générateur g et un ensemble E contenant tous les sous-ensembles de l'itemset g : $E = \{l \subseteq \mathcal{I} \mid l \subset g\}$. Puisque g est un itemset générateur, nous avons $\forall l \in E$, $\gamma(l) \subset g$. Nous en déduisons que $\psi(l) \supset \psi(g) \Longrightarrow support(l) > support(g)$. Le support d'un sous-ensemble strict s de l'itemset générateur g est supérieur au support de g. □

L'algorithme Close$^+$ peut être aisément modifier afin de déterminer les générateurs des itemsets fermés fréquents. Les ensembles G_k de k-itemsets générateurs fréquents sont alors créés itérativement de G_μ à G_1 en comparant le support de chaque k-itemset fréquent f de l'ensemble F_k avec les supports de ses sous-ensembles dans F_{k-1}. S'il existe un sous-ensemble de f dont le support est égal au support de f, alors f n'est pas un générateur et une variable booléenne *isgenerator* prend la valeur "faux". Lorsque tous les sous-ensembles de f ont été examinés, si la variable *isgenerator* possède la valeur "vrai" alors f est inséré dans l'ensemble G_k des k-générateurs fréquents. L'ensemble G_1 est initialisé avec l'ensemble F_1 contenant tous les 1-itemsets fréquents qui sont tous par définition des 1-générateurs fréquents.

5.3 Résultats expérimentaux

Nous avons implémenté les algorithmes Apriori, Close et A-Close en C++ sur diverses plate-formes Unix afin de comparer leurs performances en termes de temps d'extraction des itemsets fréquents et d'espace mémoire nécessaires à cette extraction. Ces implémentations ont été réalisées par Yves Bastide [Bas00] et utilisent toutes la même structure de données (prefix-tree), décrite dans la section 5.2.1, qui

[2]Nous appelons sous-ensemble strict d'un générateur g un itemset l tel que $l \subset g$.

permet d'améliorer les performances de l'algorithme Apriori. Les expérimentations ont été réalisées sur un PC Pentium II possédant une vitesse d'horloge de 350 Mhz et 128 Megaoctets de mémoire, fonctionnant sous le système d'exploitation Linux. Un fichier de mémoire virtuelle d'une taille de 128 Megaoctets a été utilisé, portant à 256 Megaoctets l'espace mémoire total utilisable par les programmes. Les jeux de données utilisés sont décrits dans la section 5.3.1 et les résultats de ces expérimentations sont présentés dans la section 5.3.2. Nous avons également implémenté l'algorithme Close$^+$ afin de générer les itemsets fermés fréquents et leur générateurs à partir de l'ensemble des itemsets fréquents extrait par l'algorithme Apriori. Nous avons put observer que les temps de réponse additionnels sont de l'ordre de la seconde, et donc négligeables comparés au temps d'extraction des itemsets fréquents à partir du contexte, pour tous les jeux de données utilisés.

5.3.1 Jeux de données

Nous avons utilisé lors de ces expérimentations des jeux de données fréquemment utilisés afin de comparer les performances des algorithmes de KDD. Les caractéristiques de ces jeux de données utilisées sont présentées dans la table 5.4. Les jeux

Nom	Nombre d'objets	Taille moyenne des objets	Nombre d'items
T10I4D100K	100,000	10	1,000
T20I6D100K	100,000	20	1,000
MUSHROOMS	8,416	23	127
C20D10K	10,000	20	386
C73D10K	10,000	73	2,178

TAB. 5.4 – Caractéristiques des jeux de données.

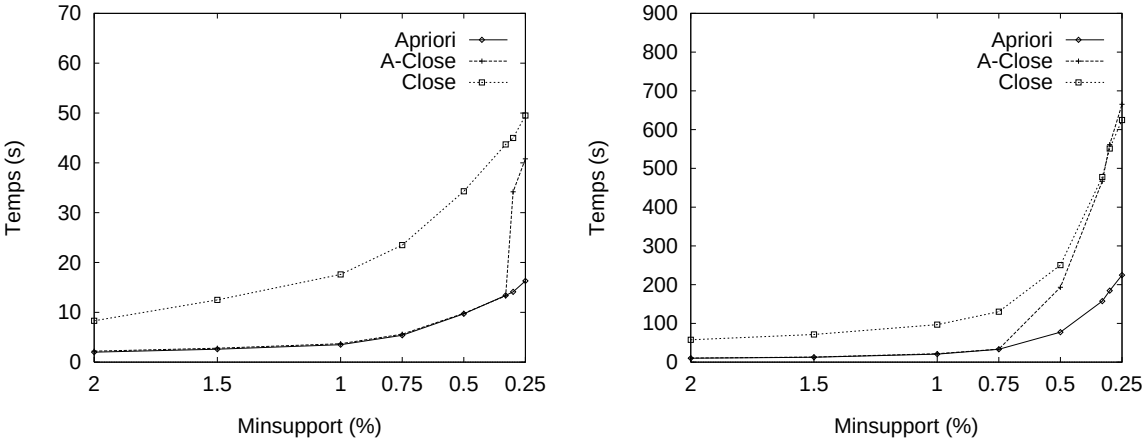
de données T10I4D100K et T20I6D100K[3] sont constitués de données synthétiques construites selon les propriétés des données de ventes. Ces jeux de données ont été générés selon la méthode présentée dans [AS94]. Ils contiennent tous les deux 100 000 objets d'une taille moyenne de 10 items pour T10I4D100K et 20 items pour T20I6D100K et d'une taille moyenne des itemsets fréquents maximaux potentiels de 4 items pour T10I4D100K et de 6 items pour T20I6D100K. Le jeu de données

[3] http ://www.almaden.ibm.com/cs/quest/syndata.html

MUSHROOMS[4] contient des informations concernant des champignons. Il est constitué de 8 416 objets d'une taille moyenne de 23 items et de 127 items correspondant aux caractéristiques des champignons au total. Les jeux de données C20D10K et C73D10K[5] sont des échantillons du fichier PUMS90KS (Public Use Microdata Samples) contenant des données du recensement du Kansas effectué en 1990. Ces deux jeux de données sont constitués des 10 000 objets correspondant aux 10 000 premières personnes recensées, chaque objet contenant 20 attributs (20 items par objets et 386 items au total) pour C20D10K et 73 attributs (73 items par objets et 2 178 items au total) pour C73D10K.

5.3.2 Résultats des expérimentations

Les temps de réponse des algorithmes Apriori, A-Close et Close pour l'extraction des itemsets fréquents dans les jeux de données synthétiques T10I4D100K et T20I6D100K sont présentés dans la figure 5.4 Les valeurs du seuil minimal de support *minsupport* utilisées sont identiques à celles utilisées dans [AS94], variant de 2% à 0,25%. Nous pouvons observer que les temps de réponse des algorithmes Apriori et



T10I4D100K T20I6D100K

FIG. 5.4 – Temps d'exécution pour les jeux de données synthétiques.

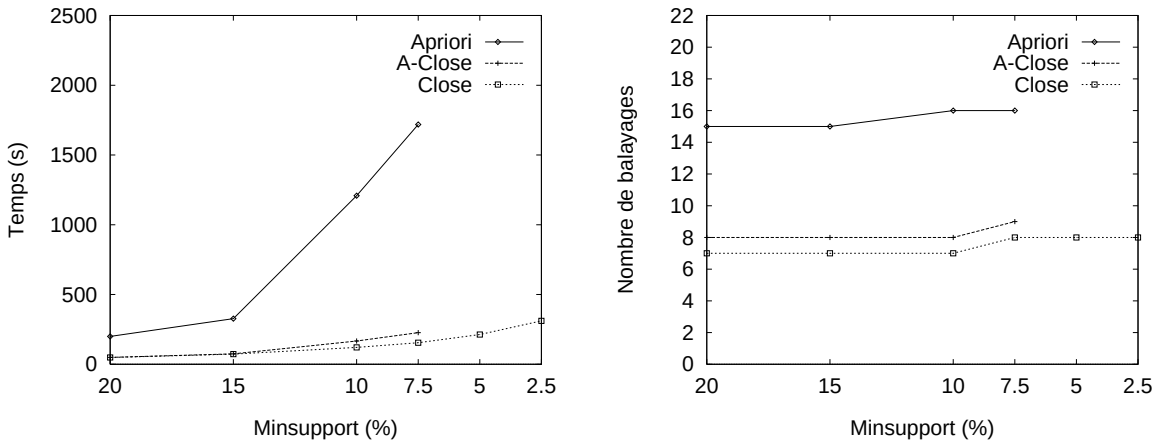
A-Close sont identiques et de 3 à 6 fois inférieurs à ceux de Close pour les seuils de support de 2% à 0,33% pour T10I4D100K et de 2% à 0,75% pour T20I6D100K. Ceci

[4] `ftp ://ftp.ics.uci.edu/~cmerz/mldb.tar.Z`

[5] `ftp ://ftp2.cc.ukans.edu/pub/ippbr/census/pums/pums90ks.zip`

s'explique par le fait que les données de ventes sont éparses et faiblement corrélées et pour les exécutions correspondant à ces seuils de support, tous les itemsets fréquents sont des itemsets fermés fréquents. En conséquence, l'espace de recherche des algorithmes Close et A-Close est identique à l'espace de recherche de l'algorithme Apriori : le treillis des itemsets fermés est identique au treillis des itemsets, et les trois algorithmes réalisent le même nombre de balayages du contexte et considèrent les mêmes ensembles d'itemsets. Cette situation correspond au pire des cas pour l'algorithme Close qui réalise plus d'opérations que l'algorithme Apriori afin de déterminer les fermetures des générateurs. Les algorithmes Apriori et A-Close réalisent les mêmes opérations du fait de l'optimisation de A-Close consistant à déterminer seulement les fermetures des générateurs dont la taille est supérieure ou égale à la taille du premier générateur fréquent non fermé identifié. Tous les générateurs fréquents étant des itemsets fermés fréquents, aucun calcul de fermeture n'est réalisé par A-Close. Dans le cas des exécutions pour des seuils de support de 0,25% pour T10I4D100K et de 0,5% à 0,25% pour T20I6D100K, certains itemsets fréquents ne sont pas fermés et l'algorithme A-Close doit calculer la fermeture de certains générateurs, ce qui entraîne des temps d'exécution supplémentaires par rapport à Apriori. Malgré ces différences, les temps d'exécution des trois algorithmes, qui varient de quelques secondes à quelques minutes, restent acceptables dans tous les cas et les algorithmes Apriori et Close se comportent de manière similaire lorsque le seuil minimal de support est diminué.

Les temps de réponse des trois algorithmes pour le jeu de données MUSHROOMS ainsi que le nombre de balayages réalisés par chacun d'eux sont présentés dans la figure 5.5. Pour ce jeu de données, les temps d'exécution ainsi que le nombre de balayages du contexte nécessaires aux algorithmes Close et A-Close sont très inférieurs à ceux nécessaires à l'algorithme Apriori. Pour les temps d'exécution, le facteur de réduction varie de 5 à 11, avec des temps maximaux d'exécution de environ 28 minutes et demi pour Apriori, 3 minutes et demi pour A-Close et 2 minutes et demi pour Close pour un seuil de support de 7,5%. Les temps de réponse de l'algorithme Close sont également inférieurs dans tous les cas aux temps de réponse de l'algorithme A-Close, les différences étant tout de fois minimes comparés aux différences avec les temps d'exécution de l'algorithme Apriori. Pour tous les seuils de support utilisés, le nombre de balayages réalisés par Close et A-Close est de moitié inférieur au nombre de balayages réalisés par l'algorithme Apriori. De plus, les algorithmes

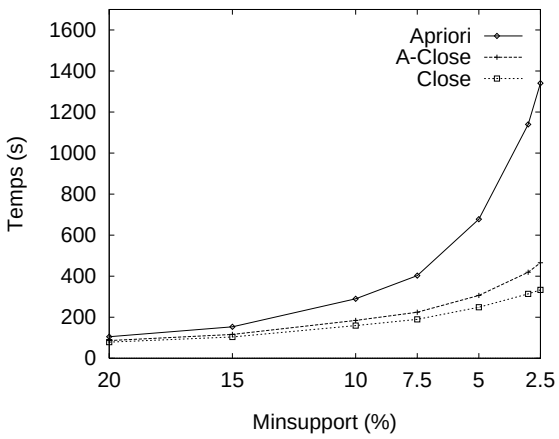


Temps d'exécution Nombre de balayages

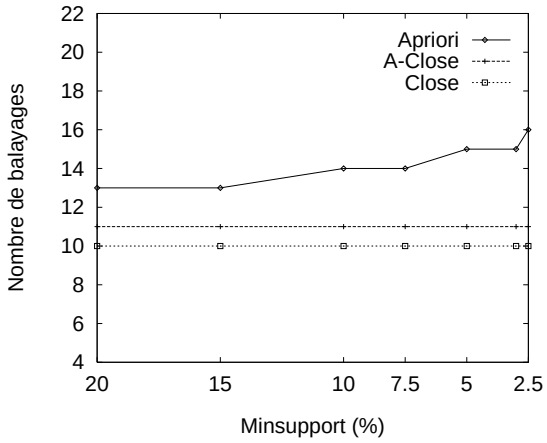
FIG. 5.5 – Temps d'exécution et nombre de balayages pour le jeu de données MUSHROOMS.

Apriori et A-Close n'ont put être exécutés pour des seuils de support inférieurs à 7,5% car leurs exécutions nécessitent un espace mémoire dépassant les capacités de la machine. Ces différences entre l'algorithme Apriori et les algorithmes Close et A-Close ont pour causes les caractéristiques des données du jeu MUSHROOMS qui sont corrélées et denses. En conséquence, le nombre d'itemsets fréquents est important et la proportion d'itemsets fermés fréquents parmi ces derniers est faible. L'espace de recherche des algorithmes Close et A-Close, qui correspond au treillis des itemsets fermés, est de taille très inférieure à l'espace de recherche de l'algorithme Apriori, qui correspond au treillis des itemsets.

Les temps d'exécution ainsi que le nombre de balayages réalisés par chacun des trois algorithmes pour les jeux de données de recensement C20D10K et C73D10K sont présentés dans les figures 5.6 et 5.7 respectivement. Pour les deux jeux de données et pour tous les seuils minimaux de support utilisés, les temps de réponse des algorithmes Close et A-Close sont nettement inférieurs à ceux de l'algorithme Apriori. De même, les nombres de balayages du contexte réalisés par Close et A-Close sont toujours inférieurs au nombre de balayages réalisés par Apriori, avec un rapport entre ces nombres variant de deux tiers à la moitié. Pour le jeu de données C20D10K, comme pour MUSHROOMS, les différences entre les temps de réponses des algorithmes Close et Apriori se mesurent en minutes ou en dizaines de minutes selon le seuil minimal de support choisi, contrairement aux temps de

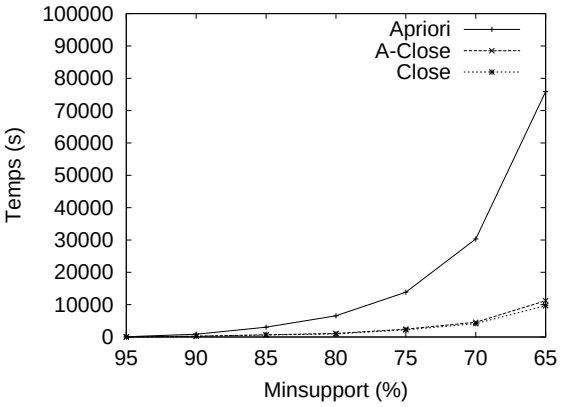


Temps d'exécution

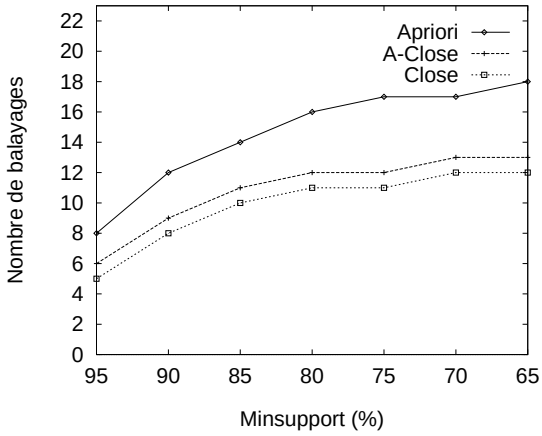


Nombre de balayages

FIG. 5.6 – Temps d'exécution et nombre de balayages pour le jeu de données C20D10K.



Temps d'exécution



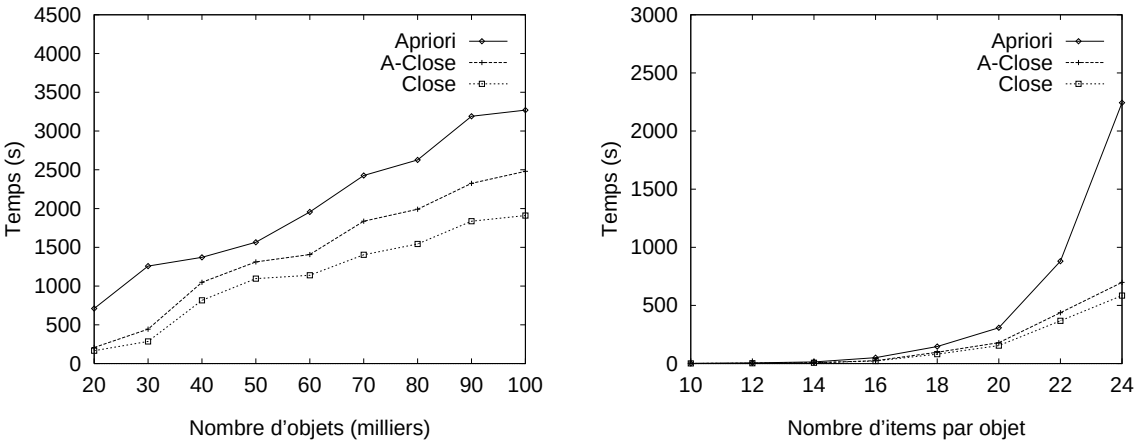
Nombre de balayages

FIG. 5.7 – Temps d'exécution et nombre de balayages pour le jeu de données C73D10K.

réponses obtenus pour les jeux de données synthétiques. De même, les temps de réponse de l'algorithme Close sont inférieurs aux temps de réponse de l'algorithme A-Close. De plus, les algorithmes Apriori et A-Close ne peuvent être exécutés sur ce jeu de données pour des seuils minimaux de support inférieurs à 2,5% car ils dépassent alors la limite de 256 Mégaoctets correspondant à la quantité maximale de mémoire utilisable par les programmes. Pour le jeu de données C73D10K, les différences entre les temps de réponses de l'algorithme Apriori et des algorithmes Close et A-Close

peuvent se mesurer en dizaines de minutes ou en heures. Pour ce jeu de données, les algorithmes Apriori et A-Close ne peuvent être exécutés pour des seuils minimaux de support inférieurs à 65% car ils dépassent alors la limite de mémoire utilisable par les programmes. Les raisons de ces différences sont identiques à celles évoquées pour le jeu de données MUSHROOMS : un nombre important d'itemsets sont fréquents parmi lesquels les itemsets fermés fréquents représentent une faible proportion. Les données de ces deux jeux possèdent les caractéristiques communes aux données statistiques, elles sont fortement corrélées et denses.

Finalement, nous avons évalué de manière expérimentale le comportement des trois algorithmes lorsque le nombre d'objets et le nombre d'items du contexte d'extraction sont augmentés. Nous avons réalisé deux séries d'expérimentation sur des jeux de données construits à partir du fichier de données de recensement duquel proviennent les jeux C20D10K et C73D10K utilisés précédemment. Les temps de réponse des trois algorithmes pour ces deux séries d'expérimentations sont présentés dans la figure 5.8. Pour la première série, nous avons fixé la taille des objets à 20



Augmentation du nombre d'objets Augmentation du nombre d'items

FIG. 5.8 – Propriétés d'augmentation des algorithmes pour les données de recensements.

attributs (20 items par objets et 386 items au total) et le seuil minimal de support à 10%, et nous avons augmenté le nombre d'objets de 20 000 à 100 000. Nous pouvons observer que les temps de réponse des trois algorithmes augmente de manière linéaire dans le nombre d'objets du contexte. Pour la seconde série, nous avons fixé le nombre d'objets à 10 000 et le seuil minimal de support de 10%, et nous avons fait

varier la taille des objets de 10 items (281 items au total) à 24 (408 items au total). L'algorithme Apriori ne peut être exécuté pour des tailles des objets supérieures à 24. Nous pouvons constater que si les temps de réponse des trois algorithmes augmentent de manière exponentielle dans la taille des objets, les algorithmes Close et A-Close possèdent des propriétés de croissance très avantageuses par rapport à l'algorithme Apriori.

5.3.3 Choix de l'algorithme

Le problème de la détermination de l'approche et donc de l'algorithme qui sera la plus efficace selon le jeu de données utilisé est complexe. Dans le cas de données non corrélées et éparses, l'approche de l'algorithme Apriori, c'est à dire le parcours du treillis des itemsets, permet d'obtenir de meilleurs temps de réponse. Toutefois, l'extraction des itemsets fréquents à partir de ce type de données avec les algorithmes Close et A-Close, par le parcours du treillis des itemsets fermés, donne des temps de réponses acceptables. Dans le cas de données corrélées et/ou denses, l'algorithme Apriori donne des temps de réponse très importants et bien supérieurs à ceux des algorithmes Close et A-Close. La nature des données qui constituent le jeu de données permet donc d'évaluer à priori l'algorithme le plus efficace, les propriétés de corrélation et de densité de nombreux types de données ayant été étudiées dans la littérature. Ainsi, il a été établi que les données de ventes de supermarchés sont éparses et faiblement corrélées [AMS+96, BMUT97] car dans ce type de données, le nombre moyen d'items par objets est faible devant le nombre total d'items et chaque item n'est contenu que dans un petit nombre d'objets. Il a également été établi qu'une importante proportion des bases de données réelles sont constituées de données corrélées ou denses. Ce sont les données statistiques [BMS97, SW85] et spatiales [KH95], les collections de textes [SBM98] et d'images [OO98], les historiques d'accès Internet [CMS97], etc.

Il est également possible d'évaluer l'efficacité des algorithmes Close et A-Close par rapport à l'algorithme Apriori pour un jeu de données en déterminant la proportion de 1-itemsets fréquents qui sont fermés. Cette proportion constitue un indicateur efficace de la proportion totale d'itemsets fréquents qui sont fermés et si presque tous les 1-itemsets fréquents sont fermés, il est probable que presque tous les itemsets fréquents sont fermés et donc les calculs de fermetures réalisés par Close et A-Close

entraîneront des temps de réponse plus faibles pour Apriori. Dans le cas contraire, peu d'itemsets fréquents sont fermés et la diminution du nombre d'itérations et de candidats considérés par Close et A-Close entraînera de meilleurs résultats de ces algorithme comparés à Apriori. La proportion de 1-itemsets fréquents qui sont fermés peut être calculée après la première itération de l'algorithme Close ou bien après la seconde itération de l'algorithme A-Close.

5.4 Discussion

Plusieurs algorithmes d'extraction des ensembles fermés à partir d'une relation binaire finie ont été proposés dans la littérature. Parmi ces derniers, nous pouvons citer l'algorithme de Bordat [Bor86], l'algorithme de Carpineto [CR93] et l'algorithme de Ganter [GR91] implémenté dans ConImp [Bur98] qui est le plus général puisqu'il permet de calculer les ensembles fermés quelque soit l'opérateur de fermeture utilisé et le plus efficace parmi ces trois algorithmes. Toutefois, ces algorithmes ne sont pas applicables dans le contexte du KDD car ils ne permettent de calculer les ensembles fermés dans des temps raisonnables que pour des contextes d'extraction comportant au plus quelques dizaines d'attributs (items) et quelques centaines d'objets. Les contextes d'extraction du KDD sont constitués pour la plupart de plusieurs centaines à plusieurs milliers d'attributs et de plusieurs dizaines de milliers à plusieurs millions d'objets. Ces algorithmes nécessitant dans le meilleur des cas autant de balayages du contexte d'extraction qu'il existe d'ensembles fermés dans le contexte, ils ne peuvent être utilisés dans la cadre du KDD. De plus, ils ne prennent pas en considération le support des itemsets afin de limiter l'espace de recherche et déterminent tous les itemsets fermés dont une proportion importante possèdent de faibles supports et ne sont donc pas significatifs pour les applications du KDD. Les algorithmes Aprem et Impec proposés dans [TPBL99, Tao00] permettent également de calculer les ensembles fermés quelque soit l'opérateur de fermeture utilisé et il a été démontré que ces algorithmes sont plus efficaces que l'algorithme de Ganter en termes de temps d'exécution et d'applicabilité puisqu'il peuvent être utilisés pour des relations de tailles plus importantes. Toutefois ces algorithmes ont étés développés dans un cadre autre que le KDD afin de résoudre des problèmes de nature différente de ceux liés au domaine du KDD et leur application à ce domaine pose des problèmes de performances.

Les méthodes utilisées par les algorithmes Close et A-Close possèdent plusieurs autres avantages importants par rapport aux méthodes utilisées par les algorithmes de Bordat, de Carpineto et de Ganter. Elles permettent d'une part de limiter autant que faire se peut le nombre de balayages du contexte nécessaires : les algorithmes Close et A-Close nécessitent un nombre de balayages égal à la taille du plus grand des générateurs des itemsets fermés fréquents, augmenté de un pour A-Close. D'autre part, les générateurs étant minimaux au sens de l'inclusion et donc de la taille, ces deux algorithmes limitent autant que possible les coûts en temps CPU des opérations sur les itemsets, plus particulièrement les tests d'inclusions et les intersections, qui dépendent directement de la taille des itemsets.

Les expérimentations démontrent que dans de nombreux cas les algorithmes Close et A-Close permettent de diminuer les temps d'extraction des itemsets fréquents et, pour l'algorithme Close, l'espace mémoire nécessaire à l'extraction ce qui en augmente le champ d'application. Ceci est plus particulièrement vrai pour les données denses et/ou corrélées qui représentent une part importante des bases de données existantes. De plus, les itemsets fermés fréquents permettent de générer l'ensemble des règles d'association valides ou bien des bases pour les règles d'association valides. Ces bases, qui sont l'objet du chapitre suivant, sont des ensembles de taille réduite ne contenant aucune règle redondante ce qui permet d'améliorer la pertinence des règles extraites. Comme il a été mentionné précédemment, l'extraction des itemsets fermés fréquents peut être utilisée afin de résoudre certains autres problèmes du KDD :

- Dans le cadre de la découverte des séries chronologiques, pour laquelle l'algorithme Apriori d'extraction des itemsets fréquents a été utilisé [AS95]. L'approche consiste a extraire les itemsets fréquents du contexte, transformer le contexte d'extraction de manière à associer à chaque objet la liste des itemsets fréquents qu'il contient et générer les séries chronologiques qui sont des ensembles ordonnés d'itemsets. L'approche que nous proposons pour l'extraction des itemsets fréquents, basée sur l'extraction des itemsets fermés fréquents, peut donc être utilisée dans ce cadre et les algorithmes Close et A-Close peuvent permettre de réduire les temps d'extraction des séries chronologiques.
- Dans [Wai98, Wai99], une méthode efficace de clustering basée sur la construction et la maintenance incrémentale du treillis de concepts fréquents est proposée. Les clusters (classes) d'objets sont alors les ensembles fermés fréquents

d'objets qui sont des regroupements maximaux d'objets contenant un même ensemble d'items et dont la taille est au moins égale à seuil minimal (*minsupport*) : ce sont les images des itemsets fermés fréquents par la fermeture γ' de la connexion de Galois. Cette méthode a été implémentée dans un système de gestion de bases de données orientées objets. Elle utilise une représentation du contexte d'extraction par listes de OID, identique à celle présentée dans la section 2.3.2, et la construction et la maintenance du treillis sont réalisées à l'aide des opérateurs définis sur le treillis de concepts fréquents présentés dans [WTL97, WTL98]. Le treillis de concepts fréquents peut être construit de manière efficace à partir des itemsets fermés fréquents, qui sont les composantes intentions des concepts fréquents, en déterminant leurs fermetures selon l'opérateur γ' qui constituent les composantes extensions des concepts fréquents. Cette détermination peut être réalisée après l'extraction des itemsets fermés fréquents en réalisant un balayage supplémentaire du contexte d'extraction, ou bien de manière plus efficace pendant le processus d'extraction en modifiant légèrement les algorithmes Close ou A-Close.

– La classification peut également bénéficier des améliorations de l'extraction des itemsets fréquents apportées par notre approche. Un système de classification basé sur la construction d'un modèle de classification à partir des algorithmes d'extraction des règles d'association a été proposé dans [LHM98]. L'approche proposée consiste à extraire un sous-ensemble particulier de l'ensemble des règles d'association valides dans le contexte, appelé ensemble de *règles d'association de classes*, et construire un modèle de classification à partir de ce sous-ensemble. Cette approche semble de plus être particulièrement efficace en termes de précision du modèle construit par rapport aux approches proposées précédemment [LHM98]. Les algorithmes Close et A-Close peuvent donc être utilisés afin d'améliorer l'efficacité de l'extraction des règles d'association, mais également la qualité du modèle produit en utilisant les itemsets fermés fréquents au lieu des itemsets fréquents pour construire les règles d'association de classes. En effet, les itemsets fermés fréquents sont des regroupements maximaux d'items communs au ensembles d'objets et peuvent donc représenter des descripteurs utiles des classes des objets.

L'utilité des algorithmes Close et A-Close ne se limite donc pas à l'extraction des règles d'association, mais ils permettent également d'améliorer les temps de réponses

d'un certain nombre d'autres tâches du KDD et, pour l'algorithme Close, d'étendre l'applicabilité de ces tâches vis à vis des besoins en espace mémoire.

Chapitre 6

Génération de bases pour les règles d'association

Sommaire

6.1 Introduction

Le problème de la pertinence et de l'utilité des règles extraites est un problème majeur de l'extraction des règles d'association. Ce problème est lié au nombre de règles d'association extraites qui est en général très important et à la présence d'une forte proportion de règles redondantes, c'est à dire de règles convoyant la même information, parmi celles-ci. La notion de règles redondantes dépend des caractéristiques des règles qui sont prises en considération et deux definitions différentes sont présentées dans la section 6.2 et la section 6.3. Si le problème de la visualisation d'un nombre relativement important de règles peut être simplifié par l'utilisation de systèmes de visualisation tels que le système Rule Visualizer proposé par Klemettinen et al. [KMR+94], le problème de la suppression des règles d'association redondantes nécessite d'autres solutions. De plus, les règles d'association redondantes représentant pour certains type de données la majorité des règles extraites, leur suppression permet de réduire considérablement le nombre de règles à gérer lors de la visualisation. La solution que nous proposons consiste à générer des bases pour les règles d'association qui sont des ensembles de tailles réduites ne contenant aucune règle redondante. Le but est de limiter l'extraction aux règles d'association les plus informatives, c'est à dire les plus générales et, éventuellement, dont les mesures de précision sont les plus élevées parmi toutes les règles valides, du point de vue de l'utilisateur.

Exemple 6.1 Afin d'illustrer le problème des règles d'association redondantes, nous présentons neuf règles d'association extraites du jeu de données Mushrooms. Ces neuf règles possèdent un support et une confiance identiques de 51% et 54% respectivement :

1) lamelles libres → comestible
2) lamelles libres → comestible, voile partiel
3) lamelles libres → comestible, voile blanc
4) lamelles libres → comestible, voile partiel, voile blanc
5) lamelles libres, voile partiel → comestible
6) lamelles libres, voile partiel → comestible, voile blanc
7) lamelles libres, voile blanc → comestible
8) lamelles libres, voile blanc → comestible, voile partiel
9) lamelles libres, voile partiel, voile blanc → comestible

Il est évident que les règles 1 à 3 et 5 à 9 sont redondantes par rapport à la règle 4 puisque, du point de vue de l'utilisateur, ces 8 règles n'apportent aucune information supplémentaire par rapport à la règle 4 qui est la plus générale. Afin d'améliorer la pertinence et l'utilité des règles extraites, il est souhaitable que seule cette dernière règle soit extraite et présentée à l'utilisateur.

Règles d'association exactes et approximatives

Dans la suite, nous distinguons deux types de règles d'association : les *règles d'association exactes*, notées $l_1 \Rightarrow (l_2 \setminus l_1)$, dont la confiance est égale à 1 et les *règles d'association approximatives*, notées $l_1 \rightarrow (l_2 \setminus l_1)$, dont la confiance est inférieure à 1. Les règles exactes sont générées à partir de deux itemsets fréquents l_1 et l_2 tels que $l_1 \subset l_2$ et qui possèdent des supports identiques : $support(l_1) = support(l_2)$.

Exemple 6.2 Les règles d'association exactes extraites du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 sont représentées dans la table 6.1.

Règle exacte	Support	Règle exacte	Support
A ⇒ C	3/6	BC ⇒ E	4/6
B ⇒ E	5/6	CE ⇒ B	4/6
E ⇒ B	5/6	AB ⇒ CE	2/6
AB ⇒ C	2/6	AE ⇒ BC	2/6
AB ⇒ E	2/6	ABC ⇒ E	2/6
AE ⇒ B	2/6	ABE ⇒ C	2/6
AE ⇒ C	2/6	ACE ⇒ B	2/6

TAB. 6.1 – Règles d'association exactes extraites du contexte $\mathcal{D}$ pour *minsupport* = 2/6.

Les règles approximatives sont générées à partir de deux itemsets fréquents l_1 et l_2 tels que $l_1 \subset l_2$ et qui possèdent des supports différents : $support(l_1) > support(l_2)$.

Exemple 6.3 Les règles d'association approximatives extraites du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance de 2/5 sont représentées dans la table 6.2.

Règle approximative	Support	Confiance	Règle approximative	Support	Confiance
BCE → A	2/6	2/4	A → BE	2/6	2/3
AC → BE	2/6	2/3	B → AE	2/6	2/5
BC → AE	2/6	2/4	E → AB	2/6	2/5
BE → AC	2/6	2/5	A → CE	2/6	2/3
CE → AB	2/6	2/4	C → AE	2/6	2/5
AC → B	2/6	2/3	E → AC	2/6	2/5
BC → A	2/6	2/4	B → CE	4/6	4/5
BE → A	2/6	2/5	C → BE	4/6	4/5
AC → E	2/6	2/3	E → BC	4/6	4/5
CE → A	2/6	2/4	A → B	2/6	2/3
BE → C	4/6	4/5	B → A	2/6	2/5
A → BCE	2/6	2/3	C → A	3/6	3/5
B → ACE	2/6	2/5	A → E	2/6	2/3
C → ABE	2/6	2/5	E → A	2/6	2/5
E → ABC	2/6	2/5	B → C	4/6	4/5
A → BC	2/6	2/3	C → B	4/6	4/5
B → AC	2/6	2/5	C → E	4/6	4/5
C → AB	2/6	2/5	E → C	4/6	4/5

TAB. 6.2 – Règles d'association approximatives extraites du contexte $\mathcal{D}$ pour *minsupport* = 2/6 et *minconfiance* = 2/5.

Nous différencions les règles d'association exactes et approximatives car elles possèdent des propriétés différentes par rapport aux itemsets fermés fréquents. Ces propriétés, qui permettent d'identifier les règles redondantes, permettent également d'identifier les règles les moins significatives, c'est à dire dont la confiance est la plus faible, parmi toutes les règles d'association valides.

Dans la section 6.2, nous présentons l'adaptation des bases pour les règles d'implications définies en analyse de données au cadre de l'extraction de règles d'association. Cette adaptation a été proposée dans [PBTL99a, TPBL00]. Dans la section 6.3, nous définissons de nouvelles bases pour les règles d'association qui ne représentent aucune perte d'information et contiennent les règles d'association non redondantes d'antécédents minimaux et de conséquences maximales. Les algorithmes de générations de ces bases sont présentés dans la section 6.4 et les résultats expérimentaux

dans la section 6.5. Les définitions de ces nouvelles bases ainsi que les algorithmes de génération de ces dernières ont été proposés dans [Pas00]. L'interêt de la génération des bases et des algorithmes proposés est discuté dans la section 6.6.

6.2 Adaptation des bases pour les règles d'implication

La définition de bases pour les règles d'implication entre deux ensembles d'attributs binaires a été étudiée essentiellement dans les domaines de l'analyse de données [DG86, Lux91] et de l'analyse formelle de concepts [GW99]. Nous nous sommes intéressé principalement à des bases pour les règles d'implication, qui ont été définies en utilisant les ensembles fermés avec pour objectif de minimiser autant que faire se peut le nombre de règles d'implications générées, provenant de l'analyse de données. Ce sont la base de Duquenne-Guigues pour les implications globales, définie par Duquenne et Guigues [DG86, GW99], et les bases de Luxenburger pour les implications partielles, définie par Luxenburger [Lux91]. Les implications globales sont des implications vérifiées dans tous les objets du contexte, contrairement aux implications partielles qui sont appelées « implications avec quelques contre exemples » ou « implications valides dans un sous-contexte ». Associée à chaque règle d'implication partielle, nous avons une mesure appelée *précision* de la règle définie de manière identique à la confiance. La précision des règles d'implication globales est égale à un. La base de Duquenne-Guigues et les bases de Luxenburger sont des ensembles générateurs de règles d'implication globales et partielles respectivement qui minimisent le nombre de règles qu'elles contiennent. Cela signifie qu'elles ne contiennent aucune règle d'implication redondante et qu'il est possible de déduire de ces base toutes les règles d'implications globales et toutes les règles d'implication partielles ainsi que leurs précisions. Les travaux de Duquenne et Guigues concernent les règles d'implications globales et utilisent donc l'opérateur d'inférence selon l'ensemble d'axiomes d'Armstrong sur l'ensemble des règles d'implication globales afin de définir les règles redondantes (la précision des règles n'est pas considérée). Les travaux de Luxenburger concernent les règles d'implication partielles et ce dernier « étend » l'opérateur d'inférence afin de prendre en considération les précisions des règles et donc de définir les règles d'implication partielles redondantes. La définition des règles d'implication

redondantes selon cet opérateur d'inférence étendu, c'est à dire en tenant compte des précisions des règles, est donnée dans la définition 6.1. Une règle $r : l_1 \rightarrow l_2$ de précision p est notée $r : l_1 \xrightarrow{p} l_2$ et pour une règle d'implication globale nous avons $p = 1$.

Définition 6.1 (Règles d'implication redondantes)
Soit un ensemble E de règles d'implication. Une règle $r : l_1 \xrightarrow{p} l_2 \in E$ est redondante si et seulement si $E \setminus \{r : l_1 \xrightarrow{p} l_2\} \models r : l_1 \xrightarrow{p} l_2$.

Une règle d'implication $r \in E$ est redondante si la règle r ainsi que sa précision peuvent être déduits de l'ensemble $E \setminus r$ en utilisant l'opérateur d'inférence étendu par Luxenburger.

6.2.1 Base de Duquenne-Guigues pour les implications globales

La base de Duquenne-Guigues pour les règles d'implication globales [DG86, GW99] est définie en utilisant les ensembles fermés d'attributs binaires du contexte et les ensembles pseudo-fermés d'attributs du contexte selon la fermeture de la connexion de Galois γ. Les ensembles pseudo-fermés sont définis de manière récursive en fonction de leurs sous-ensembles qui sont eux-mêmes des ensembles pseudo-fermés. Soit un contexte $\mathcal{B} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$ dans lequel $\mathcal{O}$ est un ensemble d'objets, $\mathcal{A}$ est un ensemble d'attributs binaire et $\mathcal{R}$ est une relation binaire entre $\mathcal{O}$ et $\mathcal{A}$. Un ensemble $e \subseteq \mathcal{A}$ est un ensemble pseudo-fermé s'il n'est pas fermé, c'est à dire si $\gamma(e) \neq e$, et si il contient les fermetures de tous ses sous-ensembles qui sont des ensembles pseudo-fermés. Il est nécessaire de considérer l'ensemble d'attributs vide $\varnothing$, qui est inclus dans tous les ensembles d'attributs, afin de déterminer les ensembles pseudo-fermés fréquents. Si $\varnothing$ n'est pas fermé, c'est à dire si un ensemble $e \neq \varnothing$ est en relation avec tous les objets du contexte, alors $\gamma(\varnothing) = e$ et $\varnothing$ est un ensemble pseudo-fermé. En conséquence tous les ensembles pseudo-fermés doivent contenir l'ensemble e.

La base de Duquenne-Guigues est constituée de toutes les règles d'implications dont l'antécédent est un ensemble pseudo-fermé e et la conséquence est la fermeture de e : $\gamma(e)$[1]. Si l'ensemble d'attributs $\varnothing$ n'est pas fermé, la règle $\varnothing \Rightarrow \gamma(\varnothing)$ appar-

[1]Selon cette définition, la conséquence de la règle est un sur-ensemble de l'antécédent de la

tient à cette base. Puisque tous les objets du contexte en relation avec l'ensemble d'attributs e sont par définition en relation avec l'ensemble d'attributs $\gamma(e)$, les règles de cette base possèdent une précision égale à 1 (100%). Il a été démontré dans [DLM92, GW99] que la base de Duquenne-Guigues est minimale relativement au nombre de règles d'implication qu'elle contient car il ne peut exister de base, c'est à dire d'ensemble générateur, contenant moins de règles qu'il existe d'ensembles pseudo-fermés fréquents dans le contexte.

Adaptation de la base de Duquenne-Guigues

L'adaptation de la base de Duquenne-Guigues dans le cadre des règles d'association nécessite la prise en compte du support des ensembles fermés et pseudo-fermés, de l'ensemble d'attributs $\varnothing$ et la réduction de la conséquence des règles $e \Rightarrow \gamma(e)$ à la fermeture de l'ensemble pseudo-fermé e diminuée des élément de e : $\gamma(e) \setminus e$. Nous commençons par définir les itemsets pseudo-fermés fréquents de manière récursive, c'est à dire en fonction de leurs sous-ensembles qui sont eux-mêmes des itemsets pseudo-fermés fréquents.

Définition 6.2 (Itemsets pseudo-fermés fréquents)
Soit F l'ensemble des itemsets fréquents. Un itemset $l \in F$ est un itemset pseudo-fermé fréquent s'il n'est pas fermé, c'est à dire si $\gamma(l) \neq l$, et si il contient les fermetures de tous ses sous-ensembles qui sont des itemsets pseudo-fermés fréquents. L'ensemble P des itemsets pseudo-fermés est défini par :

$$P = \{l_1 \in F \mid \gamma(l_1) \neq l_1 \ \wedge \ \forall l_2 \subset l_1 \text{ tel que } l_2 \in P \text{ nous avons } \gamma(l_2) \subset l_1\}.$$

L'ensemble vide d'item $\varnothing$ qui est fréquent par définition et est inclus dans tous les itemsets fréquents doit être considéré afin de déterminer les itemsets pseudo-fermés fréquents. Si $\varnothing$ n'est pas fermé, c'est à dire si il existe un itemset $l_1 \neq \varnothing$ contenu dans tous les objets du contexte (tel que $support(l_1) = |\mathcal{O}|$), alors $\gamma(\varnothing) = l_1$ et $\varnothing$ est un itemset pseudo-fermé fréquent. Les itemsets pseudo-fermés fréquents l_2 doivent alors tous contenir l'itemset l_1. La base de Duquenne-Guigues pour les règles d'association exactes est constituée de toutes les règles d'association exactes dont l'antécédent est un itemset pseudo-fermé fréquent l et la conséquence est la fermeture de l diminuée

règle.

des items de l : $\gamma(l) \setminus l$. Puisque l n'est pas fermé, nous avons $l \neq \gamma(l)$ et donc la conséquence de la règle est non vide : $(\gamma(l) \setminus l) \neq \varnothing$.

Définition 6.3 (Base de Duquenne-Guigues)
Soit l'ensemble PF des itemsets pseudo-fermés fréquents dans le contexte $\mathcal{B}$. La base de Duquenne-Guigues pour les règles d'association exactes est :

$$BDG = \{r : l \Rightarrow (\gamma(l) \setminus l) \mid l \in PF \ \wedge \ l \neq \varnothing\}.$$

Exemple 6.4 Les itemsets pseudo-fermés fréquents ainsi que la base de Duquenne-Guigues pour les règles d'association exactes extraits du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 sont présentés dans la table 6.3. L'itemset $\varnothing$ est fermé dans le contexte $\mathcal{D}$ et n'est donc pas un itemset pseudo-fermé fréquent. Les itemsets fréquents {A}, {B} et {E} sont des itemsets pseudo-fermés fréquents car ils ne sont pas fermés et leur seul sous-ensemble qui est l'itemset $\varnothing$ n'est pas un itemset pseudo-fermé fréquent. L'itemset {AB} n'est pas un itemset pseudo-fermé fréquent car les fermetures de ses sous-ensembles {A} et {B}, qui sont respectivement {AC} et {BE}, ne sont pas inclues dans {AB}. L'itemset {ABCE} n'est pas un itemset pseudo-fermé fréquent puisqu'il est fermé.

Itemset pseudo-fermé fréquent	Fermeture	Règle exacte	Support
{A}	{AC}	A ⇒ C	3/6
{B}	{BE}	B ⇒ E	5/6
{E}	{BE}	E ⇒ B	5/6

TAB. 6.3 – Base de Duquenne-Guigues extraite du contexte $\mathcal{D}$ pour *minsupport* = 2/6.

La base de Duquenne-Guigues est une réduction de l'ensemble des règles d'association exactes valides obtenue en minimisant autant que faire se peut le nombre de règles exactes générées sans tenir compte du support des règles. Si toutes les règles d'association exactes valides dans le contexte peuvent être déduites de cette base pour ce qui est des antécédents et des conséquences des règles, il n'en est pas de même pour leurs supports. Un algorithme de génération de la base de Duquenne-Guigues pour les règles d'association exactes est présentée dans la section 6.4.1.

6.2.2 Bases de Luxenburger pour les implications partielles

Soit un contexte $\mathcal{B} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$. Luxenburger défini dans [Lux91] une famille de bases pour les règles d'implication partielles à partir des ensembles fermés d'attributs, appelés *compréhensions*.

Base propre pour les implications partielles

Luxenburger démontre que l'ensemble des règles d'implication dont l'antécédent est un ensemble d'attributs $e \subseteq A$ fermé ($\gamma(e) = e$) et la conséquence est un ensemble d'attributs $e' \subseteq A$ fermé tels que $e \subset e'$ constitue une base pour l'ensemble des règles d'implication partielles du contexte. Ces règles sont appelées *règles d'implication partielles propres* et possèdent une précision inférieure à 1 puisque $e = \gamma(e) \subset e' = \gamma(e')$ et donc $\psi(e) \subset \psi(e')$. Cette base est appelée *base propre* pour les règles d'implication partielles et peut être représentée par un graphe dirigé dont les sommets sont les ensembles fermés et les arcs représentent les règles d'implication de la base. Ce graphe correspond à la fermeture transitive du diagramme de Hasse représentant le treillis des ensembles fermés.

Adaptant cette base au cadre des règles d'association , nous définissons la base propre pour les règles d'association approximatives. Cette base est constituée de toutes les règles de la forme $f_1 \rightarrow (f_2 \setminus f_1)$ dont l'antécédent f_1 est un itemset fermé fréquent et la conséquence est un itemset fermé fréquent f_2 sur-ensemble de f_1 (tels que $f_1 \subset f_2$) diminué des items de f_1 : $(f_2 \setminus f_1)$.

Définition 6.4 (Base propre pour les règles d'association approximatives) *Soit l'ensemble FF des itemsets fermés fréquents dans le contexte $\mathcal{B}$. La base propre pour les règles d'association approximatives est :*

$$BP = \{r : f_1 \rightarrow (f_2 \setminus f_1) \mid f_1, f_2 \in FF \ \wedge \ f_1 \subset f_2 \ \wedge \ \textit{confiance}(r) \geq \text{minconfiance}\}.$$

Nous appelons les règles de la base propre règles d'association approximatives propres.

Exemple 6.5 Le graphe de représentation de la base propre extraite du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance 2/5 est présenté dans la figure 6.1. La première valeur associée à chaque arc est le support de la règle ; la seconde valeur est la confiance de la règle. L'ensemble FF des itemsets fermés fréquents utilisé pour générer les règles de la base est présenté dans la table 4.1 (section 4.3.3).

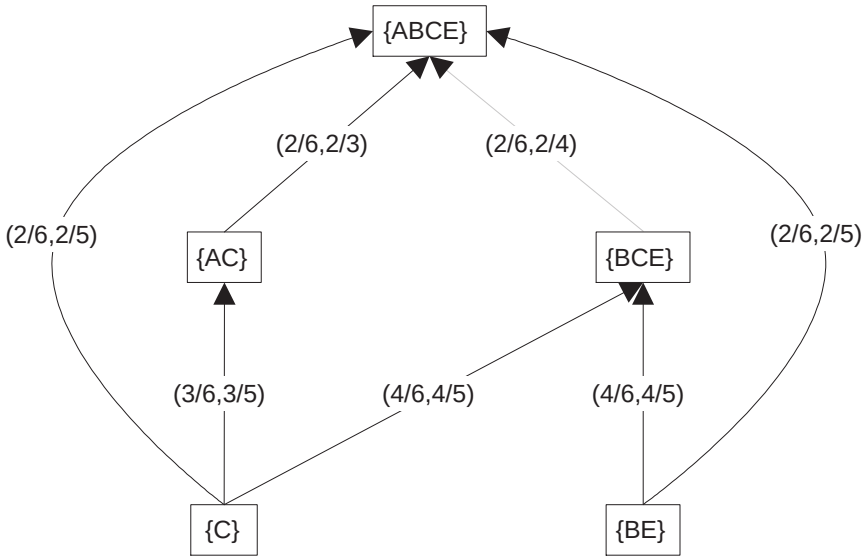


FIG. 6.1 – Graphe orienté de représentation de la base propre extraite du contexte $\mathcal{D}$ pour $minsupport = 2/6$ et $minconfiance = 2/5$.

Le nombre de règles d'association de la base propre générées à partir d'un itemset fermé fréquent maximal f correspond au nombre d'itemsets fermés fréquents qui sont des sous-ensembles de f. Toutes les règles d'association approximatives valides dans le contexte peuvent être déduites (ainsi que leurs supports et leurs confiance) de la base propre pour les règles d'association approximatives. La démonstration de cette propriété est directement déduite des propriétés introduites par Luxenburger dans [Lux91] et du fait que qu'il existe dans la base propre une règle pour chaque couple d'itemsets fermés fréquents. Les supports de toutes les règles valides peuvent être déduits des supports des règles de cette base. Un algorithme de génération de la base propre pour les règles d'association approximatives est présenté dans la section 6.4.3.

Base de couverture pour les implications partielles

Luxenburger démontre que la base propre pour les implications partielles n'est pas minimale relativement au nombre de règles qu'elle contient et défini une base qui est un sous-ensemble de la base propre. Cette base, appelée *base de couverture* pour les implications partielles, est construite selon la relation de couverture entre ensembles fermés d'attributs. Elle est constituée des règles dont l'antécédent est un

ensemble fermé f et la conséquence est un ensemble fermé f' tels que f' couvre f : $f \lessdot f'$. Cette base correspond à la réduction transitive de la base propre et peut donc être représentée sous forme d'un graphe correspondant à la réduction transitive du graphe de représentation de la base propre.

Luxenburger démontre que la base de couverture est une base pour l'ensemble des règles d'implication partielles propres et donc pour l'ensemble des règles d'implication partielles. Cette démonstration est basée sur la propriété suivante : la précision d'une règle partielle propre $e \rightarrow e'$ pour $e \subset e'$ et $e \not\lessdot e'$ est égale au produit des précisions des règles $f \rightarrow f'$ de la base de couverture entre les ensembles fermés $f \lessdot f'$ pour $f, f' \in \{e \lessdot e_1 \lessdot \ldots \lessdot e_n \lessdot e'\}$. La règle $e \rightarrow e'$ est une règle transitive du graphe représentant la base propre et les règles $f \rightarrow f'$ forment un « chemin » (succession d'arcs contigus) entre les sommets e et e' dans ce graphe. Ces règles forment également une succession d'arcs entre les sommets e et e' dans le graphe de représentation de la base de couverture.

Adaptant cette base au cadre des règles d'association, nous définissons la base de couverture pour les règles d'association approximatives. Cette base est constituée de toutes les règles dont l'antécédent est un itemset fermé fréquent f_1 et la conséquence est un itemset fermé fréquent f_2 qui couvre f_1 (tels que $f_1 \lessdot f_2$) diminué des items de f_1 : $(f_2 \setminus f_1)$.

Définition 6.5 (Base de couverture pour les règles approximatives)
Soit l'ensemble FF des itemsets fermés fréquents dans le contexte $\mathcal{B}$. La base de couverture pour les règles d'association approximatives est :

$$BP = \{r : f_1 \rightarrow (f_2 \setminus f_1) \mid f_1, f_2 \in FF \ \wedge \ f_1 \lessdot f_2 \ \wedge \ \textit{confiance}(r) \geq \text{minconfiance}\}.$$

Exemple 6.6 Le graphe de représentation de la base de couverture extraite du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance 2/5 est présenté dans la figure 6.2.

Le nombre de règles d'association de la base de couverture générées à partir d'un itemset fermé fréquent f_2 correspond au nombre d'itemsets fermés fréquents qui sont couverts pas l'itemset f_2. C'est à dire au nombre d'itemsets fermés fréquents prédécesseurs de f_2 dans le treillis des itemsets fermés. Toutes les règles d'association approximatives propre peuvent être déduites (ainsi que leurs supports et leurs confiance) de la base de couverture pour les règles d'association approximatives. La

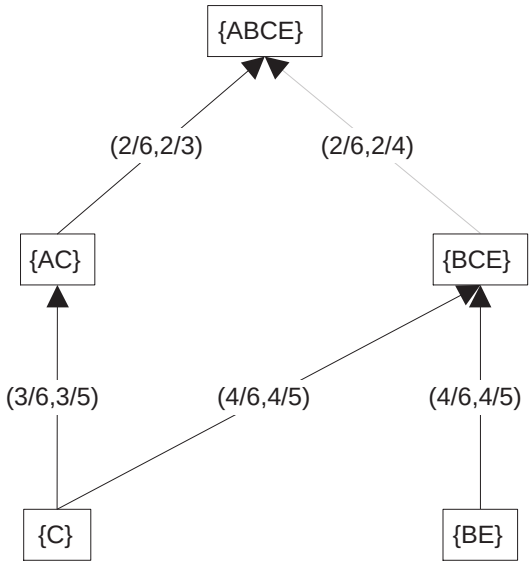


FIG. 6.2 – Graphe orienté de représentation de la base de couverture extraite du contexte $\mathcal{D}$ pour *minsupport* = 2/6 et *minconfiance* = 2/5.

démonstration de cette propriété est directe à partir des propriétés introduites par Luxenburger dans [Lux91]. Considérons par exemple la règle C → ABCE du graphe de la figure 6.2 représentant la base propre dont la confiance est 2/5 et le support est 2/6. Cette valeur de confiance est égale au produit des confiances des règles C → AC et AC → ABCE et au produit des confiances des règles C → BCE et BCE → ABCE qui forment des chemins entre les sommets {C} et {ABCE}. Le support de la règle C → ABCE est égal au support des règles AC → ABCE et BCE → ABCE.

L'algorithme de génération de la base propre pour les règles d'association approximatives présenté peut aisément être étendu pour la génération de la base de couverture pour les règles d'association approximatives. Cette extension est présentée dans la section 6.4.3.

Bases structurelles pour les implications partielles

A partir de cette propriété concernant les précisions des règles transitives, il est possible de déduire plusieurs autres bases pour les règles d'implication partielles, appelées *bases structurelles*. Ces bases sont des sous-ensembles de la base de couverture dans lesquels un unique chemin entre deux sommets e et e' (ensembles fermés) du

graphe de couverture est conservé. Selon la stratégie de sélection des chemins conservés et supprimés, plusieurs bases structurelles peuvent être définies. Luxenburger défini par exemple dans [Lux91] une base structurelle, appelée *base arborescente*, dont la représentation sous forme de graphe est un arbre couvrant maximal.

Il est possible d'étendre l'algorithme de génération de la base propre pour les règles d'association approximatives, présenté dans la section 6.4.3, pour la génération des bases structurelles pour les règles d'association approximatives. Cette extension est discutée brièvement dans la section 6.4.3.

6.3 Définition de nouvelles bases pour les règles d'association

Une règle d'association est une règle d'implication entre deux itemsets à laquelle sont associées des mesures de précision de la règle dans le contexte d'extraction. Ces mesures sont le support, qui peut être vu comme une mesure d'utilité, et la confiance (identique à la précision des règles d'implication) qui peut être vue comme une mesure de pertinence de la règle. Dans le cas des règles d'implication, le support des règles n'est pas pris en considération. Cette mesure apporte une information importante pour les règles d'association puisque elle définie la portée de la règle dans le contexte et donc l'importance de la connaissance qu'elle apporte à l'utilisateur. Afin d'étendre la définition des règles d'implication redondantes au règles d'association, nous considérons l'opérateur d'inférence tenant compte des supports et des confiances (identiques à la précision) des règles d'association. Dans la définition 6.6, nous caractérisons les règles d'association redondantes en utilisant cet opérateur. Une règle $r : l_1 \rightarrow l_2$ de support s et de confiance (précision) c est notée $r : l_1 \xrightarrow{s,c} l_2$.

Définition 6.6 (Règles d'association redondantes)
Soit un ensemble E de règles d'association. Une règle $r : l_1 \xrightarrow{s,c} l_2 \in E$ est redondante si et seulement si $E \setminus \{r : l_1 \xrightarrow{s,c} l_2\} \models r : l_1 \xrightarrow{s,c} l_2$.

Une règle d'association $r \in E$ est redondante si la règle r peut être déduite ainsi que son support s et sa confiance c de l'ensemble $E \setminus r$. Considérant la définition 6.6, les bases de Luxenburger (base propre, base de couverture et bases structurelles) sont également des bases pour les règles d'association approximatives. En effet $\forall r : l_1 \xrightarrow{s,c}$

$l_2 \in \mathcal{AR}$ telle que $c < 1$, nous avons $BP \models r$ et $BC \models r$. La base de Duquenne-Guigues ne permet pas de déduire les supports de toutes les règles d'association exactes valides et n'est donc pas un ensemble générateur pour ces règles : cette base représente donc une perte d'information pour l'utilisateur. En effet $\exists r : l_1 \xrightarrow{s,c} l_2 \in \mathcal{AR}$ telle que $c = 1$ et $BDG \not\models r$. De plus, la base de Duquenne-Guigues pour les règles d'association exactes et les bases de Luxenburger pour les règles d'association approximatives ne sont pas les bases les plus informatives pour l'utilisateur. En effet, ces bases ne sont pas constituées des *règles d'association non redondantes minimales*, c'est à dire des règles d'association non redondantes d'antécédent minimal et de conséquence maximale.

Comme nous l'avons vu dans l'exemple 6.1, il est souhaitable que seules les règles d'association non redondantes minimales, qui sont les règles les plus utiles et les plus pertinentes, soient extraites et présentées à l'utilisateur. Une règle d'association est redondante si elle convoie la même information ou une information moins générale que l'information convoyée par une autre règle de même utilité et de même pertinence. Nous définissons donc les règles d'association non redondantes minimales en tenant compte des supports et des confiances des règles d'association. Une règle d'association r est non redondante minimale s'il n'existe pas une autre règle d'association r' possédant le même support et la même confiance, dont l'antécédent est un sous-ensemble de l'antécédent de r et la conséquence est un sur-ensemble de la conséquence de r.

Définition 6.7 (Règles d'association non redondantes minimales)
Soit l'ensemble $\mathcal{AR}$ des règles d'association extraites du contexte. Une règle d'association $r : l_1 \rightarrow l_2 \in \mathcal{AR}$ est non redondante minimale s'il n'existe pas de règle d'association $r' : l'_1 \rightarrow l'_2 \in \mathcal{AR}$ telle que support(r) = support(r'), confiance(r) = confiance(r'), $l'_1 \subset l_1$ *et* $l_2 \subset l'_2$.

Dans la section 6.3.1, nous caractérisons les règles d'association exactes non redondantes minimales en utilisant les itemsets fermés fréquents et leurs générateurs. Dans la section 6.3.2, nous caractérisons les règles d'association approximatives non redondantes minimales en utilisant les itemsets fermés fréquents et leurs générateurs. À partir de ces caractérisations, nous définissons ensuite la *base générique* pour les règles d'association exactes et la *base informative* pour les règles d'association approximatives ainsi que la *réduction transitive de la base informative*.

6.3.1 Base générique pour les règles d'association exactes

La propriété 6.1 caractérise les règles d'association exactes en fonction des itemsets fermés fréquents. Nous en déduisons que les règles d'association exactes sont les règles entre les sur-ensembles stricts d'un itemset fermé fréquent f_1 et les sous-ensembles d'un itemset fermé fréquent f_2 pour $f_1 \lessdot f_2$. Les itemsets fréquents ainsi caractérisés, qui forment l'intervalle $]f_1, f_2]$[2], possèdent tous un support égal au support de f_2. Toutes les règles d'association entre deux itemsets de l'intervalle $]f_1, f_2]$ possèdent donc un même support égal au support de f_2 et une même confiance égale à 1.

Propriété 6.1 *Les règles d'association exactes* $l_1 \Rightarrow (l_2 \setminus l_1)$ *sont des règles entre deux itemsets fréquents* l_1 *et* l_2 *dont les fermetures sont identiques :* $\gamma(l_1) = \gamma(l_2)$.

Preuve. Puisque $support(l_1) = support(l_2)$ et selon la propriété 4.3, nous en déduisons que $support(\gamma(l_1)) = support(\gamma(l_2))$ et puisque $l_1 \subset l_2$, nous en concluons que $\gamma(l_1) = \gamma(l_2)$. □

Soit l'ensemble G_{f_2} des générateurs de l'itemset fermé fréquent f_2. Il est évident que les règles exactes entre deux itemsets fréquents de l'intervalle $]f_1, f_2]$ sont les règles exactes entre deux itemsets fréquents des intervalles $[g, f_2]$ pour tous les générateurs $g \in G_{f_2}$ puisque nous avons $]f_1, f_2] = \bigcup_{g \in G_{f_2}} [g, f_2]$. Pour chaque intervalle $[g, f_2]$, la règle $g \Rightarrow (g \setminus f_2)$ est la règle d'association exacte non redondante d'antécédent minimal et de conséquence maximale parmi les règles entre deux itemsets de cet intervalle. Nous en concluons que les règles de la forme $g \Rightarrow (f \setminus g)$ entre les générateurs $g \in G_{f_2}$ et l'itemset fermé fréquent f_2 sont les règles d'antécédents minimaux et de conséquences maximales parmi les règles entre deux itemsets de l'intervalle $]f_1, f_2]$. Nous généralisons cette propriété à l'ensemble des itemsets fermés fréquents et nous définissons ainsi la base générique constituée de toutes les règles d'association exactes non redondantes selon la définition 6.6 d'antécédents minimaux et de conséquences maximales.

[2] L'intervalle $]f_1, f_2]$ est défini comme l'ensemble $I = \{l \subseteq \mathcal{I} \mid f_1 \subset l \subseteq f_2\}$. L'intervalle]{A},{ABC}] par exemple est l'ensemble {{AB}, {AC}, {ABC}}.

Définition 6.8 (Base générique pour les règles d'association exactes)
Soit l'ensemble FF des itemsets fermés fréquents extraits du contexte. Soit la collection d'ensembles G_f qui associent à chaque itemset fermé fréquent f les générateurs de f. La base générique pour les règles d'association exactes est :

$$BG = \{r : g \Rightarrow (f \setminus g) \mid f \in FF \ \wedge \ g \in G_f \ \wedge \ g \neq f\}.$$

La condition $g \neq f$ est nécessaire car les règles entre un générateur g d'un itemset fermé fréquent f tel que $g = f$ sont de la forme $g \Rightarrow \varnothing$ et ne n'appartiennent pas à l'ensemble des règles d'association valides (règles non informatives).

Exemple 6.7 La base générique pour les règles d'association exactes extraite du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 est présentée dans la table 6.4.

Générateur	Fermeture	Règle exacte	Support
{A}	{AC}	A ⇒ C	3/6
{B}	{BE}	B ⇒ E	5/6
{C}	{C}		
{E}	{BE}	E ⇒ B	5/6
{AB}	{ABCE}	AB ⇒ CE	2/6
{AE}	{ABCE}	AE ⇒ BC	2/6
{BC}	{BCE}	BC ⇒ E	4/6
{CE}	{BCE}	CE ⇒ B	4/6

TAB. 6.4 – Base générique extraite du contexte $\mathcal{D}$ pour *minsupport* = 2/6.

La base générique, qui contient plus de règles que la base de Duquenne-Guigues, est la base minimale (en nombre de règles) pour les règles d'association exactes sans perte d'information. Toutefois, toutes les règles d'association exactes valides dans le contexte peuvent être déduites ainsi que leurs supports à partir des règles de cette base. Cette propriété est établie par le théorème 6.1.

Théorème 6.1 (Base pour les règles d'association exactes)
La base générique est une base pour l'ensemble des règles d'association exactes valides.

Preuve. Soit une règle d'association exacte valide $r : l_1 \Rightarrow (l_2 \backslash l_1)$ entre deux itemsets fréquents l_1 et l_2. Nous avons donc nécessairement $l_1 \subset l_2$ et puisque $confiance(r) = 1$ nous avons également $support(l_1) = support(l_2)$. D'après le lemme 4.3, nous en déduisons que $\gamma(l_1) = \gamma(l_2) = f$. L'itemset f est un itemset fermé fréquent $f \in FF$ et il existe une règle $r' : g \Rightarrow (f \setminus g) \in BG$ telle que g est un générateur de f pour lequel $g \subseteq l_1$ et $g \subset l_2$. Nous démontrons que la règle r et son support peuvent être déduits de la règle r' et son support. Puisque $g \subseteq l_1 \subset l_2 \subseteq f$, la règle r peut être reconstruite à partir de la règle r'. De $\gamma(l_1) = \gamma(l_2) = f$, nous déduisons que $support(r) = support(l_2) = support(\gamma(l_2)) = support(f) = support(r')$. □

6.3.2 Bases informatives pour les règles d'association approximatives

La propriété 6.2 caractérise les règles d'association approximatives en fonction des itemsets fermés fréquents. De cette propriété, nous déduisons une définition des ensembles de règles d'association approximatives qui possèdent les mêmes supports et les mêmes confiances. Soit quatre itemsets fermés fréquents f_1, f_1', f_2 et f_2' tels que $f_1 \lessdot f_1' \subset f_2' \lessdot f_2$. Les règles $l_1 \rightarrow (l_2 \setminus l_1)$ entre un itemset l_1 appartenant à l'intervalle $]f_1, f_1']$ et un itemset l_2 de l'intervalle $]f_2, f_2']$ possèdent des supports et des confiances identiques. Leur support est égal au support de f_2' et leur confiance est égale au rapport $support(f_1')/support(f_2')$.

Propriété 6.2 *Les règles d'association approximatives $l_1 \rightarrow (l_2 \setminus l_1)$ sont des règles entre deux itemsets fréquents l_1 et l_2 tel que la fermeture de l_1 est un sous-ensemble de la fermeture de l_2 : $\gamma(l_1) \subset \gamma(l_2)$.*

Preuve. Puisque $support(l_1) > support(l_2)$ et selon la propriété 4.3, nous avons $support(\gamma(l_1)) > support(\gamma(l_2))$. Puisque $l_1 \subset l_2$ nous en concluons que $\gamma(l_1) \subset \gamma(l_2)$. □

Soit l'ensemble $G_{f_1'}$ des générateurs de l'itemset fermé fréquent f_1'. Les règles entre les générateurs $g \in G_{f_1'}$ et l'itemset fermé fréquent f_2' de la forme $g \Rightarrow (f_2' \backslash g)$ sont les règles non redondantes d'antécédents minimaux et de conséquences maximales parmi les règles entre un itemset de l'intervalle $]f_1, f_1']$ et un itemset de l'intervalle $]f_2, f_2']$. En effet, les générateurs g sont les itemsets minimaux dont la fermeture est f_1', ce

qui signifie que les antécédents g de ces règles sont minimaux, et les conséquences $(f_2' \setminus g)$ sont maximales puisque f_2' est l'itemset maximal de l'intervalle $]f_2, f_2']$. Nous généralisons cette propriété à l'ensemble des itemsets fermés fréquents et nous définissons ainsi la base informative constituée de toutes les règles d'association approximatives non redondantes selon la définition 6.6 d'antécédents minimaux et de conséquences maximales.

Définition 6.9 (Base informative pour les règles approximatives)
Soit l'ensemble FF des itemsets fermés fréquents et l'ensemble G de leurs générateurs extraits du contexte. La base informative pour les règles d'association approximatives est :

$$BI = \{r : g \rightarrow (f \setminus g) \mid f \in FF \wedge g \in G \wedge \gamma(g) \subset f \wedge \textit{confiance}(r) \geq \text{minconfiance}\}.$$

Exemple 6.8 La base informative pour les règles d'association approximatives extraite du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 est présentée dans la table 6.5.

Générateur	Fermeture	Sur-ensemble fermé	Règle approximative	Support	Confiance
{A}	{AC}	{ABCE}	A → BCE	2/6	2/3
{B}	{BE}	{BCE}	B → CE	4/6	4/5
{B}	{BE}	{ABCE}	B → ACE	2/6	2/5
{C}	{C}	{AC}	C → A	3/6	3/5
{C}	{C}	{BCE}	C → BE	4/6	4/5
{C}	{C}	{ABCE}	C → ABE	2/6	2/5
{E}	{BE}	{BCE}	E → BC	4/6	4/5
{E}	{BE}	{ABCE}	E → ABC	2/6	2/5
{AB}	{ABCE}				
{AE}	{ABCE}				
{BC}	{BCE}	{ABCE}	BC → AE	2/6	2/4
{CE}	{BCE}	{ABCE}	CE → AB	2/6	2/4

TAB. 6.5 – Base informative extraite du contexte $\mathcal{D}$ pour $\textit{minsupport} = 2/6$.

Le théorème 6.2 démontre que toutes les règles d'association approximatives valides dans le contexte peuvent être déduites ainsi que leurs supports et leurs confiance à partir des règles de cette base.

Théorème 6.2 (Base pour les règles d'association approximatives)
La base informative est une base pour l'ensemble des règles d'association approximatives valides.

Preuve. Soit une règle d'association approximative valide $r : l_1 \rightarrow (l_2 \setminus l_1)$ entre deux itemsets fréquents l_1 et l_2. Nous avons donc nécessairement $l_1 \subset l_2$ et puisque $confiance(r) < 1$ nous avons également $\gamma(l_1) \subset \gamma(l_2)$. Quels que soient les itemsets fréquents l_1 et l_2, il existe un générateur g_1 d'un itemset fermé fréquent f_1 tels que $g_1 \subset l_1 \subseteq \gamma(l_1) = f_1$ et un générateur g_2 d'un itemset fermé fréquent f_2 tels que $g_2 \subset l_2 \subseteq \gamma(l_2) = f_2$. Puisque $l_1 \subset l_2$, nous avons $l_1 \subseteq f_1 \subset l_2 \subseteq f_2$ et la règle $r' : g_1 \rightarrow (f_2 \setminus g_1)$ appartient à la base informative BI. Nous démontrons que la règle r, son support et sa confiance peuvent être déduits de la règle r', son support et sa confiance. Puisque $g_1 \subset l_1 \subseteq f_2 \subset g_2 \subset l_2 \subseteq f_2$, l'antécédent et la conséquence de r peuvent être reconstruits à partir de la règle r'. De plus, nous avons $\gamma(l_2) = f_2$ et donc $support(r) = support(l_2) = support(f_2) = support(r')$. Puisque $g_1 \subset l_1 \subseteq f_1$, nous avons $support(g_1) = support(l_1)$ et nous en déduisons donc que :

$$confiance(r) = \frac{support(l_1)}{support(l_2)} = \frac{support(g_1)}{support(f_2)} = confiance(r'). \quad \square$$

Nous pouvons déduire de la définition de la base informative pour les règles d'association approximative une autre base qui est la réduction transitive de la base informative. En effet, les règles de la base informative de la forme $r : g \rightarrow (f \setminus g)$ telles que f est un itemset fermé fréquent et g est un générateur fréquent avec $\gamma(g) \subset f$ et $\gamma(g) \not\subset f$ sont des règles transitives. La réduction transitive de la base informative est donc constituée des règles de la forme $r : g \rightarrow (f \setminus g)$ pour un itemset fermé fréquent f et un générateur fréquent g tel que $\gamma(g) \lessdot f$.

Définition 6.10 (Réduction transitive de la base informative)
Soit l'ensemble FF des itemsets fermés fréquents et l'ensemble G de leurs générateurs extraits du contexte. La réduction transitive de la base informative pour les règles d'association approximatives est :

$$RI = \{r : g \rightarrow (f \setminus g) \mid f \in FF \wedge g \in G \wedge \gamma(g) \lessdot f \wedge confiance(r) \geq \text{minconfiance}\}.$$

Cette réduction est équivalente à la réduction transitive de la base propre pour les règles d'association approximatives qui définie la base de couverture. Elle constitue

une base pour l'ensemble des règles d'association approximatives valides car il est possible de déduire toutes les règles transitives de la base informative ainsi que leurs supports et leurs confiances à partir des règles non transitives. Cette réduction transitive permet de diminuer le nombre de règles extraites en conservant les règles dont la confiance est la plus élevée puisque les règles transitives $g \rightarrow (f \setminus g)$ pour $\gamma(g) \not< f$ possèdent des confiances inférieures aux règles non transitives $g' \rightarrow (f' \setminus g')$ qui forment un chemin entre g et f.

Exemple 6.9 La réduction transitive de la base informative pour les règles d'association approximatives extraite du contexte $\mathcal{D}$ pour un seuil minimal de support de 2/6 est présentée dans la table 6.6.

Générateur	Fermeture	Sur-ensemble fermé	Règle approximative	Support	Confiance
{A}	{AC}	{ABCE}	A → BCE	2/6	2/3
{B}	{BE}	{BCE}	B → CE	4/6	4/5
{C}	{C}	{AC}	C → A	3/6	3/5
{C}	{C}	{BCE}	C → BE	4/6	4/5
{E}	{BE}	{BCE}	E → BC	4/6	4/5
{AB}	{ABCE}				
{AE}	{ABCE}				
{BC}	{BCE}	{ABCE}	BC → AE	2/6	2/4
{CE}	{BCE}	{ABCE}	CE → AB	2/6	2/4

TAB. 6.6 – Réduction transitive de la base informative extraite du contexte $\mathcal{D}$ pour $minsupport = 2/6$.

6.4 Algorithmes de génération des bases pour les règles d'association

Dans cette section sont présentés les algorithmes de génération de la base de Duquenne-Guigues et de la base générique pour les règles d'association exactes, ainsi que les algorithmes de génération des bases de Luxenburger (base propre, base de couverture, bases structurelles) et de la base informative et sa réduction transitive pour les règles d'association approximatives.

6.4.1 Base de Duquenne-Guigues

Nous proposons un algorithme nommé Gen-BDG de génération de la base de Duquenne-Guigues pour les règles d'association exactes à partir des itemsets fréquents, parmi lesquels sont identifiés les itemsets pseudo-fermés fréquents, et des itemsets fermés fréquents. Les itemsets pseudo-fermés fréquents sont identifiés en utilisant les règles déjà générées dans la base selon la méthode proposée par Ganter et Duquenne [GW99] pour identifier les ensembles pseudo-fermés. Le pseudo-code de l'algorithme est présenté dans l'algorithme 6.1. Les notations utilisées sont présentées dans la table 6.7.

F_k	Ensemble de k-itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.
Γ_k	Ensemble de k-itemsets fermés fréquents. Chaque élément de cet ensemble possède deux champs : *fermé* et *support*.
CPF_k	Ensemble de k-itemsets pseudo-fermés fréquents potentiels. Chaque élément de cet ensemble possède un champ : *itemset*.
BDG	Ensemble des règles d'association exactes de la base de Duquenne-Guigues.
cumul	Union des fermetures des sous-ensembles pseudo-fermés du k-itemset pseudo-fermé fréquent candidat f considéré.

TAB. 6.7 – Notations utilisées dans l'algorithme Gen-BDG.

L'algorithme commence par initialiser l'ensemble BDG avec l'ensemble vide (ligne 1). Il détermine ensuite si l'itemset $\varnothing$ est un itemset pseudo-fermé fréquent en déterminant s'il existe un 1-itemset fréquent l dont le support est égal au nombre d'objets du contexte (ligne 2). Si l'itemset l existe, l'itemset $\varnothing$ est un itemset pseudo-fermé fréquent et la règle $\varnothing \Rightarrow \gamma(l)$ est insérée dans l'ensemble BDG (ligne 3). La fermeture $\gamma(l)$ de l'itemset fréquent l, qui est identique à la fermeture de $\varnothing$, est le plus petit itemset fermé fréquent contenant l. Cet itemset $\gamma(l)$ est identifié rapidement en parcourant les ensembles Γ_k dans l'ordre des valeurs de k croissantes de l'ensemble Γ_1 jusqu'à l'ensemble $\Gamma_{|\gamma(l)|}$ contenant $\gamma(l)$.

Ensuite, les ensembles F_k de k-itemsets fréquents sont parcourus successivement de l'ensemble F_1 jusqu'à l'ensemble $F_{\mu-1}$ (lignes 4 à 14). Durant une itération k, l'ensemble CPF_k de k-itemsets pseudo-fermés fréquents candidats est initialisé avec

les k-itemsets fréquents $f \in F_k$, qui ne sont pas des k-itemsets fermés fréquents : $f \notin \Gamma_k$ (ligne 5). Ensuite, chacun de ces k-itemsets pseudo-fermés fréquents candidats f est examiné afin de déterminer s'il est pseudo-fermé (lignes 6 à 13). Pour cela, l'union des fermetures des sous-ensembles pseudo-fermés de l'itemset f est calculée dans l'itemset *cumul* (lignes 7 à 10). Ces sous-ensembles pseudo-fermés sont les antécédents p des règles $p \Rightarrow (\gamma(p) \setminus p)$ de la base de Duquenne-Guigues (dans BDG) pour $p \subset f$ et leur fermeture et l'union de l'antécédent et de la conséquence de la règle : $p \cup (\gamma(p) \setminus p)$. Si par exemple les règles déjà découvertes sont les règles $\varnothing \Rightarrow$ B et A $\Rightarrow$ C, la fermeture *cumul* pour l'itemset {AE} aura la valeur {ABCE}. Si l'itemset *cumul* est inclus dans l'itemset f, alors f est un itemset pseudo-fermé fréquent et la règle $r : f \Rightarrow (\gamma(f) \setminus f)$ de support $f.support$ est insérée dans l'ensemble BDG (lignes 11 et 12). Les itérations cessent lorsque tous les ensembles F_k de k-itemsets fréquents de taille $k \leq \mu - 1$ ont été considérés et l'ensemble BDG retourné par l'algorithme contient toutes les règles de la base de Duquenne-Guigues (théorème 6.3).

Remarque 6.1 Si l'itemset $\varnothing$ est pseudo-fermé, la règle $\varnothing \Rightarrow \gamma(\varnothing)$ est générée dans l'ensemble BDG afin d'assurer la correction de la génération des autres règles. Toutefois, cette règle n'apporte aucune information supplémentaire par rapport à l'itemset $\gamma(\varnothing)$ dont le support est par définition 1 (100%).

Théorème 6.3 (L'algorithme Gen-BDG est correct)
L'algorithme Gen-BDG génère toutes les règles d'association exactes de la base de Duquenne-Guigues.

Preuve. Nous démontrons que tous les itemsets pseudo-fermés fréquents sont déterminés par l'algorithme. Par définition, les k-itemsets pseudo-fermés fréquents sont des k-itemsets fréquents qui ne sont pas des k-itemsets fermés fréquents. L'ensemble CPF des k-itemsets pseudo-fermés fréquents candidats est donc un sur-ensemble de l'ensemble des k-itemsets pseudo-fermés fréquents. Les candidats f de l'ensemble CPF à partir desquels la règle exacte $f \Rightarrow (\gamma(f) \setminus f)$ n'est pas générée dans BDG sont ceux pour lesquels la fermeture *cumul* n'est pas inclue dans f. Si tel est le cas, il existe une règle $p \Rightarrow (\gamma(p) \setminus p)$ dans BDG pour laquelle p est un pseudo-fermé fréquent sous-ensemble de f et la fermeture $\gamma(p)$ n'est pas inclue dans f. L'itemset f n'est donc pas un itemset pseudo-fermé et la règle $f \Rightarrow (\gamma(f) \setminus f)$ n'appartient

ALG. 6.1 Génération de la base de Duquenne-Guigues avec Gen-BDG.

Entrée : ensembles F_k des k-itemsets fréquents ; ensembles Γ_k des k-itemsets fermés fréquents ;

Sortie : ensemble BDG des règles d'association exactes de la base de Duquenne-Guigues ;

1) $BDG \leftarrow \varnothing$;
2) **si** $(\exists l \in F_1 \mid support(l) = |\mathcal{O}|)$
3) **alors** $BDG \leftarrow BDG \cup \{r : \varnothing \Rightarrow \gamma(l)\}$;
4) **pour chaque** ensemble F_k pour $k < \mu$ **faire**
5) $\quad CPF_k \leftarrow F_k \setminus \Gamma_k$;
6) $\quad$ **pour chaque** itemset $f \in CPF_k$ **faire**
7) $\quad\quad cumul \leftarrow \varnothing$;
8) $\quad\quad$ **pour chaque** règle $r : p \Rightarrow (\gamma(p) \setminus p)$ in BDG **faire**
9) $\quad\quad\quad$ **si** $(p \subseteq f)$ **alors** $cumul \leftarrow cumul \cup \gamma(p)$;
10) $\quad\quad$ **fin pour**
11) $\quad\quad$ **si** $(cumul \subset f)$
12) $\quad\quad$ **alors** $BDG \leftarrow BDG \cup \{r : f \Rightarrow (\gamma(f) \setminus f), f.support\}$;
13) $\quad$ **fin pour**
14) **fin pour**
15) **retourner** BDG ;

pas à la base de Duquenne-Guigues. Les μ-itemsets fréquents étant des itemsets fermés fréquents ils ne sont pas pseudo-fermés et il n'est pas nécessaire de considérer l'ensemble $F_\mu = \Gamma_\mu$. Pour tous les itemsets pseudo-fermés fréquents f, les règles $f \Rightarrow (\gamma(f) \setminus f)$ sont insérées dans l'ensemble BDG qui correspond donc à la base de Duquenne-Guigues caractérisée par la définition 6.3. □

Exemple 6.10 La figure 6.3 représente la génération de la base de Duquenne-Guigues pour les règles d'association exactes dans le contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6.

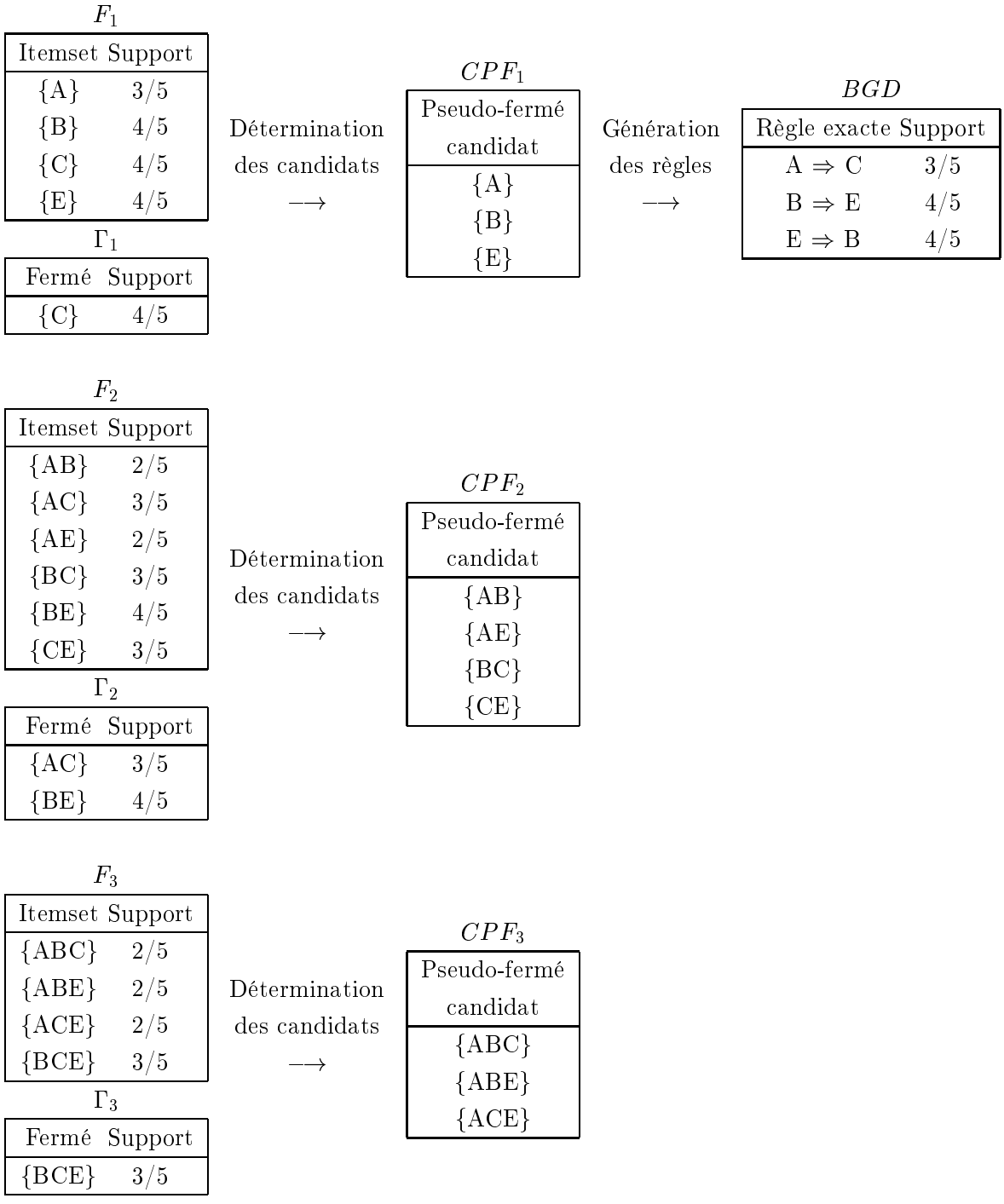


FIG. 6.3 – Génération de la base de Duquenne-Guigues dans le contexte $\mathcal{D}$ avec Gen-BDG pour *minsupport* = 2/6.

6.4.2 Base générique

Nous proposons un algorithme nommé Gen-BG de construction de la base générique pour les règles d'association exactes à partir de l'ensemble des itemsets fermés fréquents et de leurs générateurs. Le pseudo-code de l'algorithme est présenté dans l'algorithme 6.2. Les notations utilisées sont présentées dans la table 6.8.

FF_k	Ensemble de k-groupes fréquents des k-générateurs. Chaque élément de cet ensemble possède trois champs : *générateur*, *fermé* et *support*.
BG	Ensemble des règles d'association exactes de la base générique.

TAB. 6.8 – Notations utilisées dans l'algorithme Gen-BG.

L'algorithme commence par initialiser l'ensemble BG avec l'ensemble vide (ligne 1). Chaque ensemble FF_k de k-groupes fréquents est ensuite examiné successivement (lignes 2 à 6). Pour chaque k-générateur $g \in FF_k$ de l'itemset fermé fréquent $\gamma(g)$ pour lequel g est différent de se fermeture $\gamma(g)$ (lignes 3 à 5), la règle $r : g \Rightarrow (\gamma(g) \setminus g)$, dont le support est égal au support de g et $\gamma(g)$, est insérée dans BG (ligne 4). L'algorithme retourne finalement l'ensemble BG qui contient toutes les règles exactes informatives entre un générateur et sa fermeture (ligne 7).

ALG. 6.2 Génération de la base générique avec Gen-BG.

Entrée : ensemble $FF = \bigcup FF_k$ des k-groupes fréquents des k-générateurs ;
Sortie : ensemble BG des règles d'association exactes de la base de générique ;

1) $BG \leftarrow \{\}$
2) **pour chaque** ensemble $FF_k \in FF$ **faire**
3) **pour chaque** k-générateur $g \in FF_k$ **tel que** $g \neq \gamma(g)$ **faire**
4) $BG \leftarrow BG \cup \{r : g \Rightarrow (\gamma(g) \setminus g), \gamma(g).support\}$;
5) **fin pour**
6) **fin pour**
7) **retourner** BG ;

Théorème 6.4 (L'algorithme Gen-BG est correct)
L'algorithme Gen-BG construit toutes les règles d'association exactes de la base générique.

Preuve. La correction de l'algorithme repose sur le fait que toutes les règles d'association entre un générateur g et l'itemset fermé fréquent f dont il est générateur sont examinées. Tous les ensembles FF_k sont examinés successivement et chaque k-générateurs g d'un itemset fermé fréquent $\gamma(g)$ de FF_k est considéré. La règle $g \Rightarrow (\gamma(g) \setminus g)$, dont le support est égal au support de f est insérée dans BG si g et $\gamma(g)$ sont distincts. La construction de toutes les règles d'association exactes de la base générique dans l'ensemble BG est donc assurée. □

Exemple 6.11 La figure 6.4 représente la construction de la base générique pour les règles d'association exactes dans le contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6.

FF_1

Générateur	Fermé	Support
{A}	{AC}	3/6
{B}	{BE}	5/6
{C}	{C}	5/6
{E}	{BE}	5/6

Création des candidats ⟶

AC_1

Antécédent candidat
{A}
{B}
{E}

Génération des règles ⟶

BG

Règle exacte	Support
A ⇒ C	3/6
B ⇒ E	5/6
E ⇒ B	5/6

FF_2

Générateur	Fermé	Support
{AB}	{ABCE}	2/6
{AE}	{ABCE}	2/6
{BC}	{BCE}	4/6
{CE}	{BCE}	4/6

Création des candidats ⟶

AC_2

Antécédent candidat
{AB}
{AE}
{BC}
{CE}

Génération des règles ⟶

BG

Règle exacte	Support
AB ⇒ CE	2/6
AE ⇒ BC	2/6
BC ⇒ E	4/6
CE ⇒ B	4/6

FIG. 6.4 – Construction de la base générique dans le contexte $\mathcal{D}$ avec Gen-BG pour $minsupport = 2/6$.

6.4.3 Bases de Luxenburger

Nous proposons un algorithme nommé Gen-BP de génération de la base propre pour les règles d'association approximatives à partir de l'ensemble des itemsets fermés fréquents. Le pseudo-code de l'algorithme est présenté dans l'algorithme 6.3. Les

notations utilisées sont présentées dans la table 6.9. L'extension de cet algorithme pour la génération de la base de couverture et des bases structurelles pour les règles d'association approximatives est ensuite discutée.

Γ_k	Ensembles de k-itemsets fermés fréquents. Chaque élément de cet ensemble possède deux champs : *fermé* et *support*.
BP	Ensemble des règles d'association approximatives de la base propre.

TAB. 6.9 – Notations utilisées dans l'algorithme Gen-BP.

L'algorithme commence par initialiser l'ensemble BP avec l'ensemble vide (ligne 1). Il examine ensuite successivement les itemsets fermés fréquents dans les ensembles Γ_k de l'ensemble Γ_1 à l'ensemble Γ_μ (lignes 2 à 11). Pour chaque itemset fermé fréquent f_1 ainsi examiné, l'algorithme détermine l'ensemble S des sous-ensembles fermés fréquents de f_1 en appliquant la procédure Subset aux ensembles Γ_j pour $j \leq k$ (ligne 4). Pour chaque sous-ensemble fermé fréquent $f_2 \in S$, la confiance de la règle $r : f_2 \rightarrow (f_1 \setminus f_2)$ égale au rapport entre le support de f_2 et le support de f_1 est calculé (ligne 6). Si la confiance de la règle r est supérieure ou égale au seuil minimal de confiance *minconfiance*, la règle r est insérée dans l'ensemble BP (lignes 7 et 8). L'algorithme s'arrête lorsque tous les itemsets fermés fréquents ont été examinés et l'ensemble BP contient alors toutes les règles de la base propre (théorème 6.5).

Théorème 6.5 (L'algorithme Gen-BP est correct)
L'algorithme Gen-BP génère toutes les règles d'association approximatives de la base propre.

Preuve. La correction de l'algorithme repose sur le fait que toutes les règles d'association propres générée à partir des itemsets fermés fréquents sont examinées. Pour un itemset fermé fréquent f_1, tous les itemsets fermés fréquents f_2 qui sont des sous-ensembles de f_1 sont considérés. Pour chacun d'eux la règle $f_2 \rightarrow (f_1 \setminus f_2)$ est examinée et toutes les règles d'association propres générées à partir de f_1 sont donc examinées. Cette procédure étant répétée pour chaque k-itemset fermé fréquent des ensembles Γ_k pour k variant de 1 à μ, la génération de toutes les règles d'association propres dont la confiance est supérieure ou égale au seuil minimal de confiance *minconfiance* est assurée. L'ensemble BP retourné par l'algorithme est la base propre caractérisée par la définition 6.4. □

ALG. 6.3 Génération de la base propre avec Gen-BP.

Entrée : ensembles Γ_k des k-itemsets fermés fréquents ; seuil minimal de confiance *minconfiance* ;

Sortie : ensemble BP des règles d'association approximatives de la base propre ;

1) $BP \leftarrow \{\}$
2) **pour** $(k \leftarrow 1\,;\, k \leq \mu\,;\, k{+}{+})$ **faire**
3) **pour chaque** itemset fermé fréquent $f_1 \in \Gamma_k$ **faire**
4) $S \leftarrow \text{Subsets}(\bigcup \Gamma_{j<k},\, f_1)$;
5) **pour chaque** itemset fermé fréquent $f_2 \in S$ **faire**
6) $r.confiance \leftarrow f_1.support / f_2.support$;
7) **si** $(r.confiance \geq minconfiance)$
8) **alors** $BP \leftarrow BP \cup \{r : f_2 \rightarrow (f_1 \setminus f_2), f_1.support, r.confiance\}$;
9) **fin pour**
10) **fin pour**
11) **fin pour**
12) **retourner** BP ;

Exemple 6.12 La figure 6.5 représente la génération de la base propre pour les règles d'association approximatives dans le contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance de 2/5.

Base de couverture

La base de couverture est la réduction transitive de la base propre, constituée de toutes les règles d'association propres entre deux itemsets fermés fréquents liés par la relation de couverture (successeur immédiat). Nous proposons un algorithme nommé Gen-BC de génération de la base de couverture pour les règles d'association approximatives à partir de l'ensemble des itemsets fermés fréquents. Cet algorithme est une extension de l'algorithme Gen-BP. Le pseudo-code de l'algorithme est présenté dans l'algorithme 6.4. Les notations utilisées sont présentées dans la table 6.10.

La première opération de l'algorithme initialise l'ensemble BC avec l'ensemble vide (ligne 1). L'algorithme examine ensuite successivement les itemsets fermés fréquents dans les ensembles Γ_k de l'ensemble Γ_1 à l'ensemble Γ_μ (lignes 2 à 18). Pour chaque itemset fermé fréquent f_1, l'algorithme détermine les ensembles S_j des item-

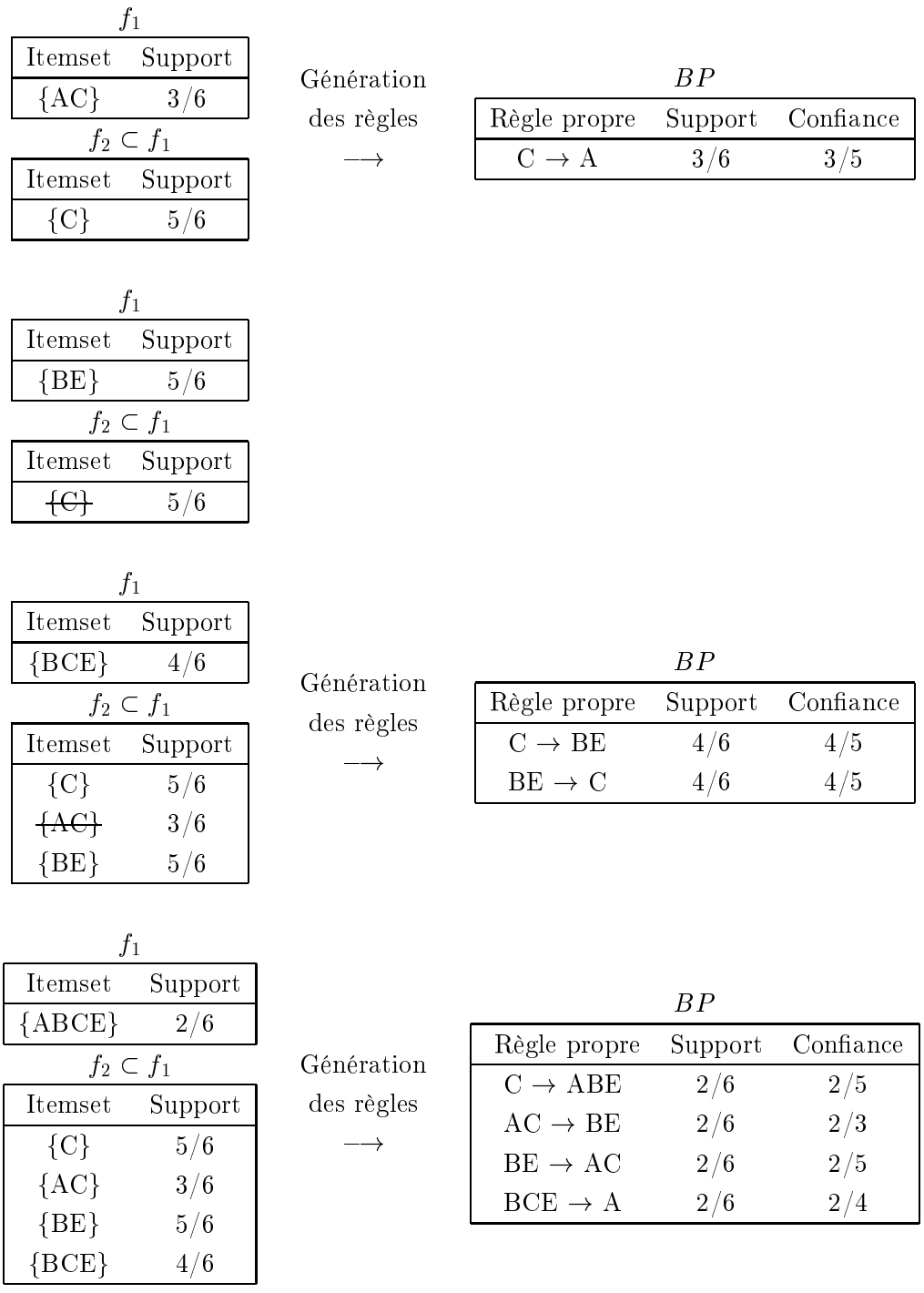


FIG. 6.5 – Génération de la base propre dans le contexte $\mathcal{D}$ avec Gen-BP pour $minsupport = 2/6$ et $minconfiance = 2/5$.

Γ_k	Ensembles de k-itemsets fermés fréquents. Chaque élément de cet ensemble possède deux champs : *fermé* et *support*.
BC	Ensemble des règles d'association approximatives de la base de couverture.

TAB. 6.10 – Notations utilisées dans l'algorithme Gen-BC.

sets fermés fréquents qui sont des sous-ensembles de f_1 en appliquant la procédure Subset aux ensembles Γ_j pour $j < k$ (lignes 4 à 6). L'algorithme considère successivement chaque ensemble S_j pour j variant de $k-1$ à 1 (lignes 7 à 16). Pour chaque sous-ensemble fermé fréquent $f_2 \in S_j$ de f_1, la confiance de la règle $r : f_2 \rightarrow (f_1 \setminus f_2)$ égale au rapport entre le support de f_2 et le support de f_1 est calculé (ligne 9). Si la confiance de la règle r est supérieure ou égale au seuil minimal de confiance *minconfiance*, la règle r est insérée dans l'ensemble BC (lignes 10 et 11). Lorsque l'ensemble f_2 a été considéré, tous les sous-ensembles de f_2 sont supprimés des ensembles S_j (lignes 12 à 14). Il n'est pas nécessaire de considérer les sous-ensembles fermés fréquents $f_3 \subset f_2$ car les règles de la forme $f_3 \rightarrow (f_1 \setminus f_3)$ ne font pas partie de la base de couverture puisque nous avons $f_3 \subset f_2 \subset f_1 \Longrightarrow f_3 \not\lessdot f_1$. L'algorithme s'arrête lorsque tous les μ-itemsets fermés fréquents ont été examinés et l'ensemble BC contient alors toutes les règles de la base de couverture (théorème 6.6).

Théorème 6.6 (L'algorithme Gen-BC est correct)
L'algorithme Gen-BC génère toutes les règles d'association approximatives de la base de couverture.

Preuve. L'algorithme Gen-BC est correct car toutes les règles d'association propres non-transitives générée à partir des itemsets fermés fréquents liés par la relation de couverture sont examinées. Pour un itemset fermé fréquent f_1, les itemsets fermés fréquents f_2 qui sont des sous-ensembles de f_1 sont considérés dans l'ordre décroissant de leur taille. Pour le premier itemset f_2 examiné nous avons donc $f_2 \lessdot f_1$ et la règle $f_2 \rightarrow (f_1 \setminus f_2)$ est insérée dans l'ensemble BC si elle possède une confiance supérieure ou égale à *minconfiance*. Ensuite, les itemsets f_3 qui sont des sous-ensembles de f_2 sont supprimés des ensembles de sous-ensembles fermés fréquents de f_1. Ces itemsets ne sont pas liés par la relation de couverture avec f_1 car nous avons $f_3 \subset f_2 \subset f_1 \Longrightarrow f_3 \not\lessdot f_1$ et les règles de la forme $f_3 \rightarrow (f_1 \setminus f_3)$ ne font donc pas partie de la base de couverture. Toutes les règles d'association propres non-transitives générées à partir de f_1 sont donc insérées dans BC. Cette

ALG. 6.4 Génération de la base de couverture avec Gen-BC.

Entrée : ensembles Γ_k des k-itemsets fermés fréquents ; seuil minimal de confiance *minconfiance* ;

Sortie : ensemble BC des règles d'association approximatives de la base de couverture ;

1) $BC \leftarrow \varnothing$;
2) **pour** ($k \leftarrow 1$; $k \leq \mu$; k++) **faire**
3) **pour chaque** itemset fermé fréquent $f_1 \in \Gamma_k$ **faire**
4) **pour chaque** ensemble Γ_j pour $j < k$ **faire**
5) $S_j \leftarrow \text{Subsets}(\Gamma_j, f_1)$;
6) **fin pour**
7) **pour chaque** ensemble S_j pour $j = k - 1$ à $j = 1$ **faire**
8) **pour chaque** itemsets $f_2 \in S_j$ **faire**
9) $r.confiance \leftarrow f_1.support \,/\, f_2.support$;
10) **si** ($r.confiance \geq minconfiance$)
11) **alors** $BC \leftarrow BC \cup \{r : f_2 \rightarrow (f_1 \setminus f_2), f_1.support, r.confiance\}$;
12) **pour chaque** ensemble S_n pour $n = j$ à $n = 1$ **faire**
13) $S_n \leftarrow S_n \setminus \text{Subsets}(S_n, f_2)$;
14) **fin pour**
15) **fin pour**
16) **fin pour**
17) **fin pour**
18) **fin pour**
19) **retourner** BC ;

procédure étant appliquée à chaque itemset fermé fréquent f_1 des ensembles FF_k pour k variant de 1 à μ, toutes les règles d'association propres non-transitives de la base de couverture sont insérées dans l'ensemble BC. □

Exemple 6.13 La génération de la base de couverture pour les règles d'association approximatives dans le contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6 et un seuil minimal de confiance de 2/5 est représentée dans la figure 6.6.

f_1

Itemset	Support
{AC}	3/6

$f_2 \subset f_1$

Itemset	Support
{C}	5/6

Génération des règles $\longrightarrow$

BC

Règle	Support	Confiance
C $\rightarrow$ A	3/6	3/5

f_1

Itemset	Support
{BE}	5/6

$f_2 \subset f_1$

Itemset	Support
~~{C}~~	5/6

f_1

Itemset	Support
{BCE}	4/6

$f_2 \subset f_1$

Itemset	Support
{C}	5/6
~~{AC}~~	3/6
{BE}	5/6

Génération des règles $\longrightarrow$

BC

Règle	Support	Confiance
C $\rightarrow$ BE	4/6	4/5
BE $\rightarrow$ C	4/6	4/5

f_1

Itemset	Support
{ABCE}	2/6

$f_2 \subset f_1$

Itemset	Support
~~{C}~~	5/6
{AC}	3/6
~~{BE}~~	5/6
{BCE}	4/6

Génération des règles $\longrightarrow$

BC

Règle	Support	Confiance
AC $\rightarrow$ BE	2/6	2/3
BCE $\rightarrow$ A	2/6	2/4

FIG. 6.6 – Génération de la base de couverture dans le contexte $\mathcal{D}$ avec Gen-BC pour $minsupport = 2/6$ et $minconfiance = 2/5$.

Bases structurelles

Les bases structurelles sont des sous-ensembles de la base propre dans lesquels des règles sont supprimées parmi les ensembles de règles qui forment des chemins entre deux sommets identiques du graphe de représentation de la base propre. Il est nécessaire que les sommets des graphes représentant les bases structurelles couvrent l'ensemble des itemsets fermés fréquents afin qu'il soit possible de déduire toutes les règles d'association approximatives valides ainsi que leurs supports et leurs confiances à partir de ces bases.

Afin d'étendre l'algorithme de génération de la base propre pour la génération d'un base structurelle, il faut intégrer la ou les stratégies de sélection des règles (correspondant aux chemins multiples) supprimées dans l'algorithme Gen-BP. Ces diverses stratégies définissent chacune une base structurelle différente et plusieurs extensions de l'algorithme peuvent ainsi être envisagées. Certaines bases structurelles sont des sous-ensembles de la réduction transitive de la base propre, c'est à dire des sous-ensembles de la base de couverture. Dans ce cas, l'intégration des stratégies de suppression des règles définissant la base structurelle peuvent être intégrées à l'algorithme Gen-BC afin de générer ces bases.

6.4.4 Bases informatives

Nous proposons un algorithme nommé Gen-BI de construction de la base informative pour les règles d'association approximatives à partir de l'ensemble des itemsets fermés fréquents et de leurs générateurs. Le pseudo-code de l'algorithme est présenté dans l'algorithme 6.5. Les notations utilisées sont présentées dans la table 6.11.

FF_k	Ensemble de k-groupes fréquents des k-générateurs. Chaque élément de cet ensemble possède trois champs : *générateur*, *fermé* et *support*.
BI	Ensemble des règles d'association approximatives de la base informative.

TAB. 6.11 – Notations utilisées dans l'algorithme Gen-BI.

L'algorithme commence par initialiser l'ensemble BI avec l'ensemble vide (ligne 1). Chaque ensemble FF_k de k-groupes fréquents est ensuite examiné successivement dans l'ordre des valeurs de k croissantes (lignes 2 à 14). Pour chaque k-générateur

$g \in FF_k$ de l'itemset fermé fréquent $\gamma(g)$ (lignes 3 à 11), les itemsets fermés fréquents $f \in FF$ de taille supérieure à la taille de $\gamma(g)$ sont considérés successivement (lignes 4 à 10). Si l'itemset fermé fréquent f est un sur-ensemble de $\gamma(g)$, la règle $r : g \rightarrow (f \setminus g)$ est insérée dans BI si la confiance de r est supérieure ou égale au seuil minimal de confiance *minconfiance* (lignes 5 à 9). Lorsque tous les générateurs de taille inférieure à μ ont été considérés, l'algorithme retourne l'ensemble BI (ligne 13).

ALG. 6.5 Génération de la base informative avec Gen-BI.

Entrée : ensemble $FF = \bigcup FF_k$ des k-groupes fréquents des k-générateurs ; seuil minimal de confiance *minconfiance* ;

Sortie : ensemble BI des règles d'association approximatives de la base de informative ;

1) $BI \leftarrow \{\}$
2) **pour** ($k \leftarrow 1$; $k \leq \mu$-1 ; k++) **faire**
3) **pour chaque** k-générateur $g \in FF_k$ **faire**
4) **pour chaque** itemset fermé fréquent $f \in FF$ **tel que** $|f| > |\gamma(g)|$ **faire**
5) **si** ($\gamma(g) \subseteq f$) **alors faire**
6) $r.confiance \leftarrow f.support/g.support$;
7) **si** ($r.confiance \geq minconfiance$)
8) **alors** $BI \leftarrow BI \cup \{r : g \rightarrow (f \setminus g), r.confiance, f.support\}$;
9) **fin si**
10) **fin pour**
11) **fin pour**
12) **fin pour**
13) **retourner** BI ;

Théorème 6.7 (L'algorithme Gen-BI est correct)
L'algorithme Gen-BI construit toutes les règles d'association approximatives de la base informative.

Preuve. La correction de l'algorithme repose sur le fait que toutes les règles d'association entre un générateur g et les itemsets fermés fréquents f qui sont des sur-ensembles de sa fermeture $\gamma(g)$ sont examinées. Tous les ensembles FF_k sont examinés successivement et pour chaque k-générateurs g d'un itemset fermé fréquent

$\gamma(g)$ de FF_k, tous les sur-ensembles fermés fréquents f de $\gamma(g)$ sont considérés. La règle $r : g \rightarrow (f \setminus g)$, dont le support est égal au support de f est insérée dans BI si sa confiance est au moins égale à *minconfiance*. L'insertion de toutes les règles d'association approximatives de la base informative dans l'ensemble BI est donc assurée. □

Exemple 6.14 La figure 6.7 représente la construction de la base informative pour les règles d'association approximatives dans le contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6.

Réduction transitive de la base informative

La réduction transitive de la base informative est constituée des règles d'association entre un générateur et un itemset fermé fréquent qui est un succésseur immédiat de la fermeture du générateur. Nous proposons un algorithme nommé Gen-RI de génération de cette réduction à partir de l'ensemble des itemsets fermés fréquents et de leurs générateurs. Cet algorithme est une extension de l'algorithme Gen-BI. Le pseudo-code de l'algorithme est présenté dans l'algorithme 6.6. Les notations utilisées sont présentées dans la table 6.12.

FF_k	Ensemble de k-groupes fréquents des k-générateurs. Chaque élément de cet ensemble possède trois champs : *générateur*, *fermé* et *support*.
$Succ_g$	Ensemble des itemsets fermés fréquents qui sont des successeurs immédiats de la fermeture du générateur g considéré.
RI	Ensemble des règles d'association approximatives de la base informative.

TAB. 6.12 – Notations utilisées dans l'algorithme Gen-RI.

L'algorithme commence par initialiser l'ensemble RI avec l'ensemble vide (ligne 1). Chaque ensemble FF_k de k-groupes fréquents est ensuite examiné successivement dans l'ordre des valeurs de k croissantes (lignes 2 à 14). Pour chaque k-générateur $g \in FF_k$ de l'itemset fermé fréquent $\gamma(g)$ (lignes 3 à 18), l'ensemble $Succ_g$ des successeurs de la fermeture de $\gamma(g)$ est initialisé avec l'ensemble vide (ligne 4) et les ensembles S_j des j-itemsets fermés fréquents qui sont des sur-ensembles de $\gamma(g)$ pour $|\gamma(g)| < j \leq \mu$ sont constuits (lignes 5 à 7). Les ensembles S_j sont ensuite considérés dans l'ordre croissant des valeurs de j (lignes 8 à 17). Pour chaque itemset

FF_1

Générateur	Fermé	Support
{A}	{AC}	3/6
{B}	{BE}	5/6
{C}	{C}	5/6
{E}	{BE}	5/6

$FF_1 \cup FF_2$

Fermé	Support
{AC}	3/6
{BE}	5/6
{C}	5/6
{BE}	5/6
{ABCE}	2/6
{BCE}	4/6

Génération des règles $\longrightarrow$

BI

Règle	Support	Confiance
A $\rightarrow$ BCE	2/6	2/3
B $\rightarrow$ CE	4/6	4/5
B $\rightarrow$ ACE	2/6	2/5
C $\rightarrow$ A	3/6	3/5
C $\rightarrow$ BE	4/6	4/5
C $\rightarrow$ ABE	2/6	2/5
E $\rightarrow$ BC	4/6	4/5
E $\rightarrow$ ABC	2/6	2/5

FF_2

Générateur	Fermé	Support
{AB}	{ABCE}	2/6
{AE}	{ABCE}	2/6
{BC}	{BCE}	4/6
{CE}	{BCE}	4/6

FF_2

Fermé	Support
{ABCE}	2/6
{BCE}	4/6

Génération des règles $\longrightarrow$

BI

Règle	Support	Confiance
BC $\rightarrow$ AE	2/6	2/4
CE $\rightarrow$ AB	2/6	2/4

FIG. 6.7 – Construction de la base informative dans le contexte $\mathcal{D}$ avec Gen-BI pour $minsupport = 2/6$ et $minconfiance = 2/5$.

$f \in S_j$ dont aucun successeur immédiat de $\gamma(g)$ dans $Succ_g$ n'est un sous-ensemble (ligne 10), f est inséré dans $Succ_g$ (ligne 11) et la confiance de la règle $r : g \rightarrow (f \setminus g)$ est calculée (ligne 12.) Si la confiance de r est supérieure ou égale au seuil minimal de confiance *minconfiance*, la règle r est insérée dans RI (lignes 13 à 15). Lorsque

tous les générateurs de taille inférieure à μ ont été considérés, l'algorithme retourne l'ensemble RI (ligne 20).

ALG. 6.6 Génération de la réduction transitive de la base informative avec Gen-RI.

Entrée : ensemble $FF = \bigcup FF_k$ des k-groupes fréquents des k-générateurs ; seuil minimal de confiance $minconfiance$;

Sortie : ensemble RI des règles d'association approximatives de la réduction transitive de la base de informative ;

1) $RI \leftarrow \{\}$
2) **pour** ($k \leftarrow 1$; $k \leq \mu$-1 ; k++) **faire**
3) **pour chaque** k-générateur $g \in FF_k$ **faire**
4) $Succ_g \leftarrow \{\}$
5) **pour** ($j = |\gamma(g)|$; $j \leq \mu$; j++) **faire**
6) $S_j \leftarrow \{f \in FF \mid f \supset \gamma(g) \ \wedge \ |f| = j\}$
7) **fin pour**
8) **pour** ($j = |\gamma(g)|$; $j \leq \mu$; j++) **faire**
9) **pour chaque** itemset fermé fréquent $f \in S_j$ **faire**
10) **si** ($\nexists s \in Succ_g \mid s \subset f$) **alors faire**
11) $Succ_g \leftarrow Succ_g \cup f$
12) $r.confiance \leftarrow f.support/g.support$;
13) **si** ($r.confiance \geq minconfiance$)
14) **alors** $RI \leftarrow RI \cup \{r : g \rightarrow (f \setminus g), r.confiance, f.support\}$;
15) **fin si**
16) **fin pour**
17) **fin pour**
18) **fin pour**
19) **fin pour**
20) **retourner** RI ;

Théorème 6.8 (L'algorithme Gen-RI est correct)
L'algorithme Gen-RI construit toutes les règles d'association approximatives de la réduction transitive de la base informative.

Preuve. La correction de l'algorithme repose sur le fait que toutes les règles d'association entre un générateur g et les itemsets fermés fréquents f qui sont des succes-

seurs immédiat de sa fermeture $\gamma(g)$ sont examinées. Tous les ensembles FF_k sont examinés successivement et pour chaque k-générateurs g d'un itemset fermé fréquent $\gamma(g)$ de FF_k, tous les sur-ensembles fermés fréquents f de $\gamma(g)$ sont considérés dans l'ordre croissant de leur taille. L'algorithme construit au fur et à mesure l'ensemble $Succ_g$ des successeurs immédiats de $\gamma(g)$. La correction de cette construction est assurée par le fait que les sur-ensembles de $\gamma(g)$ sont considérés dans l'ordre croissant de leur taille. Si aucun itemset de l'ensemble $Succ_g$ n'est un sous-ensemble de f, alors f est un successeur immédiat de $\gamma(g)$, il est inséré dans $Succ_g$ et la règle $r : g \rightarrow (f \setminus g)$, dont le support est égal au support de f, est insérée dans RI si sa confiance est au moins égale à *minconfiance*. L'insertion de toutes les règles d'association approximatives de la réduction transitive de la base informative dans l'ensemble RI est donc assurée. □

Exemple 6.15 La figure 6.8 représente la construction de la réduction transitive de la base informative pour les règles d'association approximatives dans le contexte d'extraction $\mathcal{D}$ pour un seuil minimal de support de 2/6.

6.5 Résultats expérimentaux

Nous avons réalisé des expérimentations afin d'évaluer le nombre de règles d'association extraites dans l'ensemble des règles valides et dans les bases pour les règles exactes et approximatives. Les algorithmes utilisés pour ces expérimentations ont été implémentés en C++ et les expérimentations ont été réalisées sur la même machine que celle utilisée pour les expérimentations concernant les algorithmes d'extraction des itemsets fermés fréquents, décrite dans la section 5.3. Afin de générer l'ensemble des règles d'association valides, l'algorithme Gen-Règles décrit dans la section 3.2 a également été implémenté. Les jeux de données utilisés sont les jeux T10I4D100K, MUSHROOMS, C20D10K et C73D10K décrits dans la section 5.3.1. Les résultats obtenus pour la base de Duquenne-Guigues sont présentés dans la section 6.5.1 et pour les bases pour les règles d'association approximatives dans la section 6.5.2.

6.5.1 Bases pour les règles d'association exactes

Le nombre total de règles d'association exactes valides et le nombre de règles dans la base de Duquenne-Guigues sont présentés dans la table 6.13. Nous pouvons

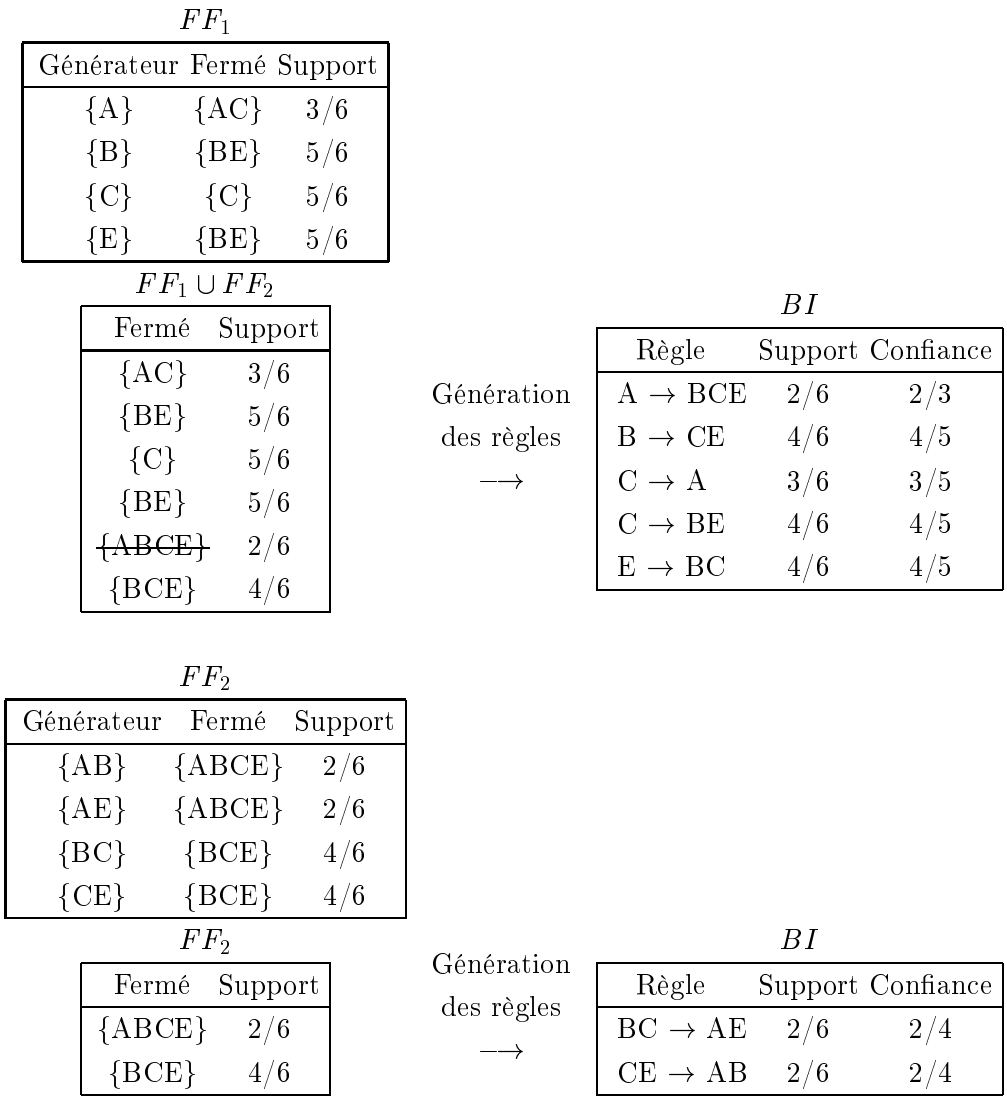


FIG. 6.8 – Construction de la réduction transitive de la base informative dans le contexte $\mathcal{D}$ avec Gen-RI pour $minsupport = 2/6$ et $minconfiance = 2/5$.

observer que aucune règle d'association exacte n'est extraite du jeu de données synthétiques T10I4D100K. En effet, puisque dans ce jeu de données et pour ce seuil minimal de support tous les itemsets fréquents sont des itemsets fermés fréquents, tous les itemsets fréquents possèdent des supports différents et tous les itemsets fermés fréquents sont eux-même leur propre générateur unique. Il n'existe donc aucune règle de la forme $l_1 \Rightarrow (l_2 \setminus l_1)$ entre deux itemsets fréquents l_1 et l_2 dont les

fermetures sont identiques : $\gamma(l_1) = \gamma(l_2)$ qui sont les règles d'association exactes valides dans le contexte.

Jeu de données	Minsupport	Règles exactes	Base de Duquenne-Guigues	Base générique
T10I4D100K	0.5%	0	0	0
MUSHROOMS	30%	7 476	69	543
C20D10K	50%	2 277	11	457
C73D10K	90%	52 035	15	1369

TAB. 6.13 – Nombre de règles d'association exactes extraites.

Pour les trois autres jeux de données, constitués de données denses et corrélées, le nombre total de règles exactes valides varie de plus de 2 000 à plus de 52 000, ce qui est considérable et rend difficile la découverte de relations intéressantes dans ces ensembles. Les bases de Duquenne-Guigues pour ces jeux de données permettent de réduire par un facteur de 100 à 3000 environ le nombre de règles exactes extraites. Le nombre de règles dans ces bases étant faible, elles apportent une connaissance utile et facilement utilisable du point de vue de l'utilisateur.

6.5.2 Bases pour les règles d'association approximatives

Le nombre total de règles d'association approximatives valides et le nombre de règles dans la bases (base propre, base de couverture, base arborescente, base informative et sa réduction transitive) sont présentés dans la table 6.14. Le nombre total de règles d'association approximatives valides est pour les quatre jeux de données très important puisqu'il varie de près de 13 000 règles pour le jeu MUSHROOMS à plus de 2 000 000 de règles pour le jeu C73D10K pour un seuil minimal de confiance de 90%. Il est donc indispensable de réduire l'ensemble des règles extraites afin de le rendre utilisable par l'utilisateur.

Pour le jeu de données T10I4D100K, la base propre et la base informative contiennent toutes les règles d'association approximatives valides car tous les itemsets fréquents sont des itemsets fermés fréquents et sont donc leurs propres et uniques générateurs. En conséquence, les règles entre deux itemsets fréquents l_1 et l_2 tels que $l_1 \subset l_2$ sont des règles entre deux itemsets fermés fréquents l_1 et $l_1 \cup l_2$ et appartiennent donc à la base propre ; ce sont également des règles entre un générateur

Jeu de données (Minsupport)	Minconfiance	Règles approximatives	Base propre	Base de couverture	Base arborescente
T10I4D100K (0.5%)	90%	16 260	16 260	3 511	916
	70%	20 419	20 419	4 004	1 058
	50%	21 686	21 686	4 191	1 140
	30%	22 952	22 952	4 519	1 367
MUSHROOMS (30%)	90%	12 911	806	563	313
	70%	37 671	2 454	968	384
	50%	56 703	3 870	1 169	410
	30%	71 412	5 727	1 260	424
C20D10K (50%)	90%	36 012	4 008	1 379	443
	70%	89 601	10 005	1 948	455
	50%	116 791	13 179	1 948	455
	30%	116 791	13 179	1 948	455
C73D10K (90%)	95%	1 606 726	23 084	4 052	939
	90%	2 053 896	32 644	4 089	941
	85%	2 053 936	32 646	4 089	941
	80%	2 053 936	32 646	4 089	941

TAB. 6.14 – Nombre de règles d'association approximatives extraites.

l_1 et un itemset fermé fréquent $l_1 \cup l_2$ et appartiennent donc à la base informative. La base de couverture et la réduction transitive de la base informative, qui sont également identiques, réduisent par un facteur de 5 en moyenne le nombre de règles approximatives extraites pour ce jeu de données ; la base arboresente réduit par un facteur de 20 en moyenne ce nombre.

Pour les jeux de données MUSHROOMS, C20D10K et C73D10K, le nombre total de règles d'association approximatives valides est en moyenne bien plus important que pour les données synthétiques. En effet, ces données étant denses et/ou corrélées, le nombre d'itemsets fréquents est bien plus élevé et il en est donc de même pour le nombre de règles approximatives valides. La proportion d'itemsets fermés fréquents parmi les itemsets fréquents étant faible, les bases pour les règles approximatives permettent de réduire considérablement le nombre de règles extraites. La base propre et la base informative, qui sont de taille sensiblement équivalentes, permettent de réduire par un facteur variant de 10 à 20 le nombre de règles extraites. La base de

Jeu de données (Minsupport)	Minconfiance	Règles approximatives	Base informative	Réduction transitive
T10I4D100K (0.5%)	90%	16 260	16 260	3 511
	70%	20 419	20 419	4 004
	50%	21 686	21 686	4 191
	30%	22 952	22 952	4 519
MUSHROOMS (30%)	90%	12 911	949	681
	70%	37 671	2 961	1 221
	50%	56 703	4 682	1 481
	30%	71 412	6 571	1 578
C20D10K (50%)	90%	36 012	4 023	1 383
	70%	89 601	10 116	1 957
	50%	116 791	13 634	1 957
	30%	116 791	13 634	1 957
C73D10K (90%)	95%	1 606 726	30 932	5 680
	90%	2 053 896	43 171	5 718
	85%	2 053 936	43 175	5 718
	80%	2 053 936	43 175	5 718

TAB. 6.14 – Nombre de règles d'association approximatives extraites.

couverture et la réduction transitive de la base informative, de tailles équivalentes également, permettent de réduire ce nombre par un facteur variant de 40 à 500. La base arborescente réduit le nombre de règles par un facteur variant de 80 à 2 000 selon le jeu de données utilisé, mais cette importante réduction entraine la suppression d'un certain nombre de règles pertinentes et utiles du point de vue de l'utilisateur, limitant ainsi l'interêt de la base arborescente par rapport aux autres bases.

L'examen des règles extraites dans chacune des bases par rapport a l'ensemble des règles valides nous a permis de vérifier qu'aucune de ces bases ne contient de règles redondantes. Considérant l'exemple présenté dans la section 6.1 concernant les neuf règles approximatives extraites du jeu de données MUSHROOMS, seule la règle 4 est générée parmi ces neuf règles dans chacune des bases. En effet, les itemsets {lamelles libres} et {lamelles libres, comestible, voile partiel, voile blanc} sont deux itemsets fermés fréquents, la règle 4 appartient donc à la base propre, et le

premier est un prédécesseur immédiat du second, ce qui signifie qu'elle appartient également à la base de couverture. Ces deux itemsets sont les seuls itemsets fermés fréquents de l'intervalle [∅, {lamelles libres, comestible, voile partiel, voile blanc}]. De plus, l'itemset fermé fréquent {lamelles libres} étant lui-même son seul et unique générateur, la règle 4 appartient également à la base informative et est donc bien la règle non redondante d'antécédent minimal et de conséquence maximale parmi ces neuf règles.

6.6 Discussion

Les définitions de la base de Duquenne-Guigues pour les implications globales et des bases de Luxenburger pour les implications partielles sont présentées dans [ZO98]. Toutefois la notion de support n'est pas prise en compte dans ces définitions et aucun algorithme de génération de ces bases pour les règles d'association n'est proposé. Un algorithme de découverte des ensembles fermés et pseudo-fermés ainsi que de la base de Duquenne-Guigues pour les règles d'implication a été proposé dans [GR91] et implémenté dans CONIMP [Bur98]. L'extraction des ensembles fermés d'attributs est basée sur l'algorithme de Ganter et l'utilisation de ce système dans le cadre de l'extraction de règles d'association pose plusieurs problèmes liés à la taille des contextes utilisés et la non prise en compte des mesures de support des règles. Aucun algorithme de découverte des bases de Luxenburger pour les règles d'implication partielles n'a été proposé.

La génération de bases pour les règles d'association présente également un fort intérêt pour la visualisation des règles extraites. En effet, le nombre réduit de règles extraites ainsi que la distinction des règles exactes et des règles approximatives facilitent la présentation des règles à l'utilisateur. De plus, du point de vue de l'utilisateur, l'absence de règles redondantes dans les bases et la génération des règles non redondantes minimales présentent un intérêt important qui est évident [KMR+94]. La construction de ces bases étant fondée sur l'utilisation des itemsets fermés fréquents, il est également possible de bénéficier des nombreux travaux concernant la visualisation réalisés dans le domaine de l'analyse formelle de concepts [SW95, Stu97].

La base générique et la base informative présentent un interêt évident car elles contiennent les règles d'association non redondantes minimales (d'antécédent minimal et de conséquence maximale). Ces bases qui ne représentent aucune perte

d'information par rapport à l'ensemble des régles d'association valides peuvent être utilisées dans de nombreux autres domaines, tels que l'analyse de données et l'analyse formelle de concepts. En effet, la définition 6.7 des règles non redondantes minimales est valide dans le cadre des règles d'implications globales et partielles entre ensembles d'attributs binaires. Les définitions 6.8 de la base générique et 6.10 de la base informative sont également valides pour les règles d'implication globales et partielles, c'est à dire sans tenir compte des supports des règles. La base générique pour les implications globales et de la base informative (et sa réduction transitive) pour les implications partielles ainsi définies, qui sont de tailles équivalentes aux bases de Duquenne-Guigues pour les implications globales et à la base propre (et la base de couverture) pour les implications partielles, présentent un interêt évident pour la caractérisation des implications entre ensembles d'attributs. La minimisation des antécédents et la maximisation des conséquences des règles de la base générique et de la base informative fournit les règles les plus informatives pour l'utilisateur. Supposons par exemple que les ensembles d'attributs $\varnothing$, {C}, {AC} et {ABCE} soient les ensembles fermés extraits. Les règles d'implication partielles C $\xrightarrow{p}$ A et AC $\xrightarrow{p'}$ BE constituent la base de couverture alors que les règles C $\xrightarrow{p}$ A et A $\xrightarrow{p'}$ BCE constituent la base informative. La règle A $\xrightarrow{p'}$ BCE est plus informative que la règle AC $\xrightarrow{p'}$ BE qui possède la même précision.

Chapitre 7

Conclusion et perspectives

Sommaire

7.1 Conclusion

Nous proposons une nouvelle approche pour l'extraction et la réduction des règles d'association dans les bases de données. Cette approche est basée sur l'utilisation de la fermeture de la connexion de Galois qui nous permet de définir le treillis des itemsets fermés duquel sont extraits les itemsets fermés fréquents. L'ensemble des itemsets fermés fréquents constitue l'élément central de notre approche. Dans cette section, nous résumons brièvement les principaux apports de nos travaux.

7.1.1 Sémantique pour le problème de l'extraction de règles d'association

Nous avons proposé une nouvelle sémantique pour l'extraction de règles d'association basée sur la connexion de Galois d'une relation binaire finie. Utilisant la connexion de Galois, nous formalisons les définitions des itemsets fréquents et les démonstrations des propriétés spécifiant que tous les sous-ensembles d'un itemset fréquent sont fréquents et que tous les sur-ensembles d'un itemset infréquent sont infréquents introduites dans [AS94, MTV94]. Ces deux propriétés constituent le fondement des algorithmes d'extraction des itemsets fréquents et d'extraction des itemsets fréquents maximaux. Utilisant la fermeture de la connexion de Galois, nous définissons les itemsets fermés qui forment le treillis des itemsets fermés et les itemsets fermés fréquents. Nous démontrons que le support d'un itemset fréquent est égal au support de l'itemset fermé fréquent minimal qui le contient et que les itemsets fréquents maximaux sont des itemsets fermés fréquents maximaux. Ces deux propriétés nous permettent de démontrer que tous les itemsets fréquents et leur supports peuvent être générés sans accéder au jeu de données à partir des itemsets fermés fréquents et leur supports. Nous proposons une nouvelle approche pour la détermination des itemsets fréquents, basée sur l'extraction préalable des itemsets fermés fréquents. Cette approche permet de limiter l'espace de recherche de la phase durant laquelle des balayages du contexte sont réalisés qui est la plus coûteuse en terme de temps d'exécution et d'espace mémoire. Nous proposons également une nouvelle approche pour l'extraction des règles d'association basée sur l'extraction des itemsets fermés fréquents qui ne nécessite pas d'extraire l'ensemble des itemsets fréquents.

7.1.2 Algorithmes d'extraction des itemsets fermés fréquents

Nous définissons les itemsets générateurs des itemsets fermés fréquents qui sont les itemsets minimaux dont la fermeture est l'itemset fermé fréquent. Utilisant ces générateurs, nous proposons deux algorithmes efficaces d'extraction par niveaux des itemsets fermés fréquents nommés Close et A-Close. Ces algorithmes réalisent un parcours en largeur du treillis des itemsets fermés pour en extraire les itemsets fermés fréquents et leurs supports. L'algorithme A-Close est basé sur les propriétés démontrant que le support d'un générateur est strictement supérieur au support de

tous ses sous-ensembles et le support d'un itemset fermé fréquent est strictement inférieur au support de tous ses sur-ensembles. Les résultats expérimentaux montrent que ces algorithmes permettent de diminuer les temps de l'extraction des itemsets fréquents dans le cas de données denses ou corrélées. Ils montrent également que l'algorithme Close permet de réduire considérablement l'espace mémoire nécessaire lors de cette phase pour ce type de données qui constituent une part importante des bases de données existantes. Nous proposons également un algorithme nommé Close$^+$ qui permet de dériver les itemsets fermés fréquents, leurs générateurs et leurs supports à partir des itemsets fréquents et leurs supports sans accéder au jeu de données. Cet algorithme permet de déterminer les itemsets fermés fréquents et leurs générateurs à partir du résultat des algorithmes d'extraction des itemsets fréquents.

7.1.3 Génération de bases pour les règles d'association

Utilisant la sémantique proposée, nous nous intéressons au problème de l'utilité et de la pertinence des règles d'association extraites. Nous adaptons les travaux de Duquenne et Guigues [DG86, GW99] et de Luxenburger [Lux91] concernant les règles d'implication entre ensembles d'attributs binaires d'une relation binaire finie au cadre de l'extraction de règles d'association. Ceci nous permet de définir la base de Duquenne-Guigues pour les règles d'association exactes, qui représente une perte d'information, ainsi que la base propre, la base de couverture, qui est la réduction transitive de la base propre, et les bases structurelles pour les règles d'association approximatives. Nous démontrons que les règles d'association de ces bases ne sont pas les règles non redondantes minimales, c'est à dire d'antécédents minimaux et de conséquences maximales. Nous définissons formellement les règles d'association exactes et approximatives ainsi que les règles d'association non redondantes minimales dont nous déduisons la base générique pour les règles d'association exactes et la base informative pour les règles d'association approximatives ainsi que sa réduction transitive qui constitue également une base. Toutes les informations convoyées par l'ensemble des règles d'association valides sont également convoyées par l'union de ces deux bases. Nous proposons des algorithmes de génération de la base de Duquenne-Guigues à partir des itemsets fréquents et des itemsets fermés fréquents et de génération de la base propre et de la base de couverture à partir des itemsets

fermés fréquents. Nous proposons également deux algorithmes de génération de la base générique et de la base informative à partir des itemsets fermés fréquents et leurs générateurs, qui sont extraits par les algorithmes Close, A-Close et Close$^+$.

7.2 Perspectives

Les résultats de nos travaux présentés dans ce mémoire offrent plusieurs perspectives de recherches ultérieures au niveau théorique et au niveau pratique. Dans cette section, nous présentons succinctement quelques unes de ces perspectives qui nous paraissent être les plus intéressantes.

7.2.1 Techniques d'implémentation et structures de données

Nous avons pu constater lors des implémentations et des expérimentations des diverses algorithmes que les structures de données et les techniques d'implémentation utilisées ont une influence particulièrement importante sur les temps d'exécution et l'espace mémoire nécessaires à l'exécution des algorithmes. L'étude des diverses techniques d'implémentation et des structures de données permettant d'améliorer les tâches de l'extraction de connaissances dans les bases de données, en fonction de leurs propriétés et des propriétés des différents types de données, nous paraît donc être particulièrement importante pour la résolution des problèmes du KDD [Bas00]. Les résultats préliminaires de cette étude suggèrent que les structures de données les plus efficaces sont les structures de bitmaps, avec les bitmaps simples et les bitmaps hiérarchiques, ainsi que les structures arborescentes, avec les arbres de hachage, les arbres de préfixes et les arbres ternaires.

7.2.2 Maintenance incrémentale de l'ensemble des itemsets fermés fréquents

La maintenance incrémentale de l'ensemble des itemsets fermés fréquents consiste en la répercution des mises à jour des données de la base de données (ajout d'un nouvel objet, d'un nouvel attribut ou de nouvelles valeurs des attributs) sur cet ensemble. Cette approche est particulièrement utile pour toutes les tâches du KDD pouvant être réalisées à partir de cet ensemble (extraction de règles d'association

et de séries chronologiques, clustering et classification supervisée) dans le cas d'applications nécessitant des exécutions fréquentes de ces tâches. Des opérateurs de construction et de maintenance incrémentale du treillis de concept fréquents, sur lequel sont appliqués des opérateurs de construction de clusters, ont été implantés dans le SGBD orienté objets O_2 [Wai99]. Les résultats expérimentaux démontrent que cette approche pose des problèmes de performances dans le cas de jeux de données de grandes tailles et le développement d'un algorithme efficace de maintenance incrémentale de l'ensemble des itemsets fermés fréquents constitue une perspective intéressante de travaux ultérieurs.

7.2.3 Règles d'association avec négation

Une autre perspective intéressante concerne le développement d'un algorithme efficace d'extraction des ensembles fréquents de littéraux spécifiant l'occurrence ou la non occurrence d'un item dans l'ensemble [Bas00]. Ces ensembles présentent deux principaux intérêts :

- L'extension du domaine d'application des règles d'association avec la génération de règles avec négation des occurrences des items, c'est à dire la prise en compte de la non occurrence des items dans les implications entre itemsets.
- L'amélioration de la mesure de la précision des règles d'association approximatives en utilisant des mesures statistiques autres que la confiance telles que la mesure de conviction définie par Brin et al. [BMUT97] par exemple.

L'extraction des ensembles fréquents de tels littéraux pose des problèmes plus importants en termes de temps d'exécution et d'espace mémoire que ceux posés par l'extraction des itemsets fréquents. Pour chaque item du contexte d'extraction, deux littéraux doivent être pris en compte, le premier correspondant à l'occurrence et le second à la non occurrence de l'item.

7.2.4 Réduction de l'ensemble des règles d'association extraites aux règles non redondantes minimales

La caractérisation des règles d'association non redondantes minimales présentée dans la définition 6.7 permet d'identifier ces règles dans l'ensemble des règles d'association valides extraites. Ceci permet d'extraire les règles d'association qui consti-

tuent la base générique et la base informative de l'ensemble des règles d'association valides extraites. Il est ainsi possible d'étendre une implémentation existante d'extraction des règles d'association ou bien d'intégrer cette méthode dans le système de visualisation afin de présenter les règles d'association non redondantes minimales à l'utilisateur. Il serait également intéressant d'étudier l'utilité de l'extraction de sous-ensemble de la base générique et de la base informative, en limitant par exemple l'extraction au règles d'association non redondantes minimales dont l'antécédent est minimal (selon la relation d'inclusion) parmi les règles possédant la même conséquence. Cette réduction des règles extraites entraîne une perte d'information, mais les règles conservées possèdent un antécédent minimal et, par construction, un support maximal parmi les règles non redondantes de même conséquence ; ce sont donc les plus informatives du point de vue de l'utilisateur.

Bibliographie

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of Data (SIGMOD'98)*, pages 94–105. ACM Press, June 1998.

[AGI+92] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifier for database mining applications. In *Proceedings of the 18th international conference on Very Large Data Bases (VLDB'92)*, pages 560–573. Morgan Kaufmann, August 1992.

[AIS93a] R. Agrawal, T. Imielinski, and A. Swami. Database mining : A performance perspective. *IEEE Transansaction on Knowledge and Data Engineering : Special issue on learning and discovery in knowledge-based databases*, 5(6) :914–925, December 1993.

[AIS93b] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of Data (SIGMOD'93)*, pages 207–216. ACM Press, May 1993.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and data mining*, pages 307–328. AAAI Press, 1996.

[AMS97] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *Proceedings of the 3rd international conference on Knowledge Discovery and Data mining (KDD'97)*, pages 115–118. AAAI Press, August 1997.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th international conference on Very Large Data Bases (VLDB'94)*, pages 478–499. Morgan Kaufmann, September 1994.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages 3–14. IEEE Computer Society Press, March 1995.

[Ass90] D. S. Associates. *The new direct marketing.* Business One Irwin, 1990.

[AY98] C. C. Aggarwal and P. S. Yu. A new framework for itemset generation. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems (PODS'98)*, pages 18–24. ACM Press, June 1998.

[BAG99] R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of the 15th International Conference on Data Engineering (ICDE'99)*, pages 188–197. IEEE Computer Society Press, March 1999.

[Bas00] Y. Bastide. *Algorithmique de data mining : techniques d'implémentation et négation.* PhD thesis, Université de Clermont-Ferrand II, 2000. En préparation.

[Bay98] R. J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of Data (SIGMOD'98)*, pages 85–93. ACM Press, June 1998.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees.* Wadsworth Publishing Company, 1984.

[Bir67] G. Birkhoff. Lattice theory. In *Colloquium Publications*, volume 25. American Mathematical Society, 1967. Third edition.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree : An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of Data (SIGMOD'90)*, pages 322–331. ACM Press, May 1990.

[BM70] M. Barbut and B. Monjardet, editors. *L'ordre et la classification.* Algèbre et combinatoire, tome II. Hachette, 1970.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets : Generalizing association rules to correlation. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of Data (SIGMOD'97)*, pages 265–276. ACM Press, May 1997.

[BMUT97] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of Data (SIGMOD'97)*, pages 255–264. ACM Press, May 1997.

[Bor86] J. P. Bordat. Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques et sciences humaines*, 96 :31–47, 1986.

[BP97] E. Baralis and G. Psaila. Designing templates for mining association rules. *Journal of Intelligent Information Systems*, 9(1) :7–32, July 1997.

[Bur98] P. Burmeister. Formal concept analysis with ConImp : Introduction to the basic features. Technical report, Technische Hochschule Darmstadt, Darmstadt, Germany, 1998. http ://www.mathematik.tu-darmstadt.de/ burmeister/ConImpIntro.ps.

[Cat91] J. Catlett. *Megainduction : Machine learning on very large databases.* PhD thesis, University of Sydney, June 1991.

[CHY96] M.-S. Chen, J. Han, and P. S. Yu. Data mining : An overview from a database perspective. *IEEE Transansaction on Knowledge and Data Engineering*, 8(6) :866–883, December 1996.

[CMS97] R. Cooley, B. Mobasher, and J. Srivastava. Web mining : Information and pattern discovery on the world wide web. In *Proceedings of the 9th International Conference on Tools with Artificial Intelligence (ICTAI'97)*, pages 558–567. IEEE Computer Society, November 1997.

[CMS99] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1) :5–32, Febuary 1999.

[CR93] C. Carpineto and G. Romano. Galois : An order-theoretic approach to conceptual clustering. In *Proceedings of the 10th International Conference on Machine Learning (ICML'90)*, pages 33–40, July 1993.

[CS96] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass) : Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and

R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, 1996.

[Czy96] A. Czyzewski. Mining knowledge in noisy audio data. In *Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96)*, pages 220–225. AAAI Press, August 1996.

[DG86] V. Duquenne and J.-L. Guigues. Famille minimale d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 24(95) :5–18, 1986.

[DL98] G. Dong and J. Li. Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. In *Proceedings of the 2nd Pacific-Asia international conference on research and development in Knowledge Discovery and Data mining (PAKDD'98)*, Lecture Notes in Computer Science, Vol. 1394, pages 72–86. Springer-Verlag, April 1998.

[DLM92] J. Demetrovics, L. Libkin, and I. B. Muchnik. Functional dependencies in relational databases : A lattice point of view. *Discrete Applied Mathematics*, 40 :155–185, 1992.

[DP94] B. A. Davey and H. A. Priestley. *Introduction to lattices and order.* Cambridge University Press, 1994. Fourth edition.

[EDV97] S. Elo-Dean and M. Viveros. Data mining the IBM official 1996 Olympics Web site. Technical report, IBM Research Center T. J. Watson, Yorktown, USA, 1997. http ://domino.watson.ibm.com/library/cyberdig.nsf/.

[EKS97] M. Ester, H.-P. Kriegel, and J. Sander. Spatial data mining : A database approach. In *Proceedings of the 5th international Symposium on advances in Spatial Databases (SSD'97)*, Lecture Notes in Computer Science, Vol. 1262, pages 47–66. Springer-Verlag, July 1997.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96)*, pages 226–231. AAAI Press, August 1996.

[EKX95] M. Ester, H.-P. Kriegel, and X. Xu. A database interface for clustering in large spatial databases. In *Proceedings of the 1st international*

conference on Knowledge Discovery and Data mining (KDD'95), pages 94–99. AAAI Press, August 1995.

[EP96] J. Elder and D. Pregibon. A statistical perspective on knowledge discovery in databases. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 83–115. AAAI Press, 1996.

[FD95] R. Feldman and I. Dagan. Knowledge discovery in textual databases. In *Proceedings of the 1st international conference on Knowledge Discovery and Data mining (KDD'95)*, pages 112–117. AAAI Press, August 1995.

[FGLS98] A. Fayet, A. Giacometti, D. Laurent, and N. Spyratos. Découverte de règles pertinentes dans les bases de données. In *Actes des 14èmes journées Bases de Données Avancées (BDA'98)*, pages 197–211, Octobre 1998.

[Fis87] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2) :139–172, 1987.

[FPSS96a] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery : An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–30. AAAI Press, 1996.

[FPSS96b] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining : Towards a unifying framework. In *Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96)*, pages 82–88. AAAI Press, August 1996.

[FPSSU96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[FYMT96] T. Fukada, S. Morishita Y. Morimoto, and T. Tokuyama. Data mining using two-dimensional optimized association rules : Scheme, algorithms, and visualization. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of Data (SIGMOD'96)*, pages 13–23. ACM Press, June 1996.

[GHQ95] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-query processing in data warehousing environments. In *Proceedings of the 21st interna-*

tional conference on Very Large Data Bases (VLDB'95), pages 358–369. Morgan Kaufmann, September 1995.

[GKMT97] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems (PODS'97)*, pages 209–216. ACM Press, May 1997.

[GM94] R. Godin and R. Missaoui. An incremental concept formation approach for learning from databases. *Theoretical Computer Science : Special issue on formal methods in databases and software engineering*, 133(2) :387–419, October 1994.

[GMA95] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence*, 11(2) :246–267, May 1995.

[GMS97] D. Gunopulos, H. Mannila, and S. Saluja. Discovering all most specific sentences by randomized algorithms. In *Proceedings of the 6th biennial International Conference on Database Theory (ICDT'97)*, Lecture Notes in Computer Science, Vol. 1186, pages 215–229. Springer-Verlag, January 1997.

[GPW98] G. Gardarin, P. Pucheral, and F. Wu. Bitmap based algorithms for mining association rules. In *Actes des 14èmes journées Bases de Données Avancées (BDA'98)*, pages 157–175, Octobre 1998.

[GR91] B. Ganter and K. Reuter. Finding all closed sets : A general approach. *Order*, 8 :283–290, 1991.

[GW99] B. Ganter and R. Wille. *Formal Concept Analysis : Mathematical foundations*. Springer, 1999.

[HCC92] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases : An attribute oriented approach. In *Proceedings of the 18th international conference on Very Large Data Bases (VLDB'92)*, pages 547–559. Morgan Kaufmann, August 1992.

[HCC93] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Transansaction on Knowledge and Data Engineering*, 5(1) :29–40, Febuary 1993.

[HCC+97] J. Han, J. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, K. Koperski G. Liu, Y. Lu, , N. Stefanovic, L. Winstone, B. Xia, O. R. Zaïane, S. Zhang, and H. Zhu. DBMiner : A system for data mining in relational databases and data warehouses. In *Proceedings of the 1997 GASCON metting of minds (GASCON'97)*, pages 249–260, November 1997. ftp ://ftp.fas.sfu.ca/pub/cs/han/kdd/cascon97.ps.

[Hec96] D. Heckerman. Bayesian networks for knowledge discovery. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 273–305. AAAI Press, 1996.

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st international conference on Very Large Data Bases (VLDB'95)*, pages 420–431. Morgan Kaufmann, September 1995.

[HF96] J. Han and Y. Fu. Exploration of the power of attribute oriented induction in data mining. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 399–421. AAAI Press, 1996.

[HFW+96] J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, and O. R. Zaïane. DBMiner : A system for mining knowledge in large relational databases. In *Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96)*, pages 250–255. AAAI Press, August 1996.

[HKM+96] K. Hätönen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen. Knowledge discovery from telecommunication network alarm databases. In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, pages 115–122. IEEE Computer Society Press, February 1996.

[HKS97] J. Han, K. Koperski, and N. Stefanovic. GeoMiner : A system prototype for spatial data mining. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of Data (SIGMOD'97)*, pages 553–556. ACM Press, May 1997.

[HRU96] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of Data (SIGMOD'96)*, pages 205–216. ACM Press, June 1996.

[HS95] M. Houtsma and A. Swami. Set-oriented mining for association rules in relational databases. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages 25–33. IEEE Computer Society Press, March 1995.

[Joh97] G. H. John. *Enhancements to the data mining process.* PhD thesis, University of Stanford, 1997.

[KH95] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Proceedings of the 4th international Symposium on advances in Spatial Databases (SSD'95)*, Lecture Notes in Computer Science, Vol. 951, pages 47–66. Springer-Verlag, August 1995.

[KHA98] K. Koperski, J. Han, and J. Adhikary. Mining knowledge in geographical data. Technical Report. ftp ://ftp.fas.sfu.ca/pub/cs/han/kdd/geo_survey98.ps, 1998.

[KK96] D. Keim and H. Kriegel. Visualization techniques for mining large databases : A comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6) :923–938, December 1996.

[KMR+94] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the 3rd international Conference on Information and Knowledge Management (CIKM'94)*, pages 401–407. ACM Press, November 1994.

[KMT97] M. Klemettinen, H. Mannila, and H. Toivonen. A data-mining methodology and its application to semi-automatic knowledge acquisition. In *Proceedings of the 8th international conference on Database and Expert systems Applications (DEXA'97)*, Lecture Notes in Computer Science, Vol. 1308, pages 670–677. Springer-Verlag, September 1997.

[LAS97] B. Lent, R. Agrawal, and R. Srikant. Discovering trends in text databases. In *Proceedings of the 3rd international conference on Knowledge*

Discovery and Data mining (KDD'97), pages 227–230. AAAI Press, August 1997.

[LHM98] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the 4th international conference on Knowledge Discovery and Data mining (KDD'98)*, pages 80–86. AAAI Press, August 1998.

[LHWC99] B. Liu, W. Hsu, K. Wang, and S. Chen. Visually aided exploration of interesting association rules. In *Proceedings of the 3nd Pacific-Asia international conference on research and development in Knowledge Discovery and Data mining (PAKDD'99)*, Lecture Notes in Computer Science, Vol. 1574, pages 380–389. Springer-Verlag, April 1999.

[Lin98] D. Lin. *Fast algorithms for discovering the maximum frequent sets*. PhD thesis, University of New York, 1998.

[Lip87] R. P. Lippmann. An introduction to computing with neural networks. *IEEE Acoustics, Speech and Signal Processing Magazine*, 4(2) :4–22, April 1987.

[LK98] D. Lin and Z. M. Kedem. Pincer-Search : A new algorithm for discovering the maximum frequent set. In *Proceedings of the 6th Biennial International Conference on Extending Database Technology (EDBT'98)*, Lecture Notes in Computer Science, Vol. 1377, pages 105–119. Springer-Verlag, March 1998.

[LSL95] H. Lu, R. Setiono, and H. Liu. NeuroRule : A connectionist approach to data mining. In *Proceedings of the 21st international conference on Very Large Data Bases (VLDB'95)*, pages 478–489. Morgan Kaufmann, September 1995.

[Lux91] M. Luxenburger. Implications partielles dans un contexte. *Mathématiques, Informatique et Sciences Humaines*, 29(113) :35–55, 1991.

[Mac98] International Business Machines. Intelligent Miner. http ://www.software.ibm.com/data/iminer/, 1998.

[Man97] H. Mannila. Methods and problems in data mining. In *Proceedings of the 6th biennial International Conference on Database Theory (ICDT'97)*, Lecture Notes in Computer Science, Vol. 1186, pages 41–55. Springer-Verlag, January 1997.

[MCPS93] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro. Systems for knowledge discovery in databases. *IEEE Transansaction on Knowledge and Data Engineering : Special issue on learning and discovery in knowledge-based databases*, 5(6) :903–913, December 1993.

[Mic83] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning : An artificial intelligence approach*, volume 1, pages 83–134. Morgan Kaufmann, 1983.

[MM93] J. Major and J. Mangano. Selecting among rules induced from a hurricane database. In *AAAI'93 Workshop on Knowledge Discovery in Databases*, pages 28–47. AAAI Press, July 1993.

[MPC96] R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *Proceedings of the 22nd international conference on Very Large Data Bases (VLDB'96)*, pages 122–133. Morgan Kaufmann, September 1996.

[MS83] R. S. Michalski and R. E. Stepp. Learning from observation : Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning : An artificial intelligence approach*, volume 1, pages 331–363. Morgan Kaufmann, 1983.

[MST94] D. Michie, D. J. Spiegelhalter, and C. C. Taylord. *Machine learning, neural, and statistical classification.* Ellis Horwood, 1994.

[MT96a] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96)*, pages 146–151. AAAI Press, August 1996.

[MT96b] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96)*, pages 189–194. AAAI Press, August 1996.

[MT97] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3) :241–258, September 1997.

[MTV94] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *AAAI'94 Workshop on Knowledge Discovery in Databases*, pages 181–192. AAAI Press, July 1994.

[MTV95] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the 1st international conference on Knowledge Discovery and Data mining (KDD'95)*, pages 210–215. AAAI Press, August 1995.

[Mue95] A. M. Mueller. Fast sequential and parallel algorithms for association rules mining : A comparison. Technical report, Faculty of the Graduate School of The University of Maryland, 1995.

[MYGS91] M. McLeach, P. Yao, M. Garg, and T. Stirtzinger. Discovery of medical diagnostic information : An overview of methods and results. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 477–490. AAAI Press, 1991.

[NH94] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th international conference on Very Large Data Bases (VLDB'94)*, pages 144–155. Morgan Kaufmann, September 1994.

[NLHP98] R. T. Ng, V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of Data (SIGMOD'98)*, pages 13–24. ACM Press, June 1998.

[OO98] C. Ordonez and E. Omiecinski. Image mining : A new approach for data mining. Technical Report GIT-CC-98-12, Georgia Institute of Technology, Atlanta, USA, 1998. ftp ://ftp.cc.gatech.edu/pub/coc/tech_reports/98/GIT-CC-98-12.ps.Z.

[Pas00] N. Pasquier. Extraction de bases pour les règles d'association à partir des itemsets fermés fréquents. In *Actes du 18ème congrès sur l'Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'2000)*, May 2000. A paraître.

[PBTL98] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Pruning closed itemset lattices for association rules. In *Actes des 14èmes journées Bases de Données Avancées (BDA'98)*, pages 177–196, Octobre 1998.

[PBTL99a] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set based discovery of small covers for association rules. In *Actes des 15èmes journées Bases de Données Avancées (BDA'99)*, pages 361–381, Octobre 1999.

[PBTL99b] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th biennial International Conference on Database Theory (ICDT'99)*, Lecture Notes in Computer Science, Vol. 1540, pages 398–416. Springer-Verlag, January 1999.

[PBTL99c] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1) :25–46, 1999.

[PCY95] J. S. Park, M.-S. Chen, and P. S. Yu. An efficient hash based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of Data (SIGMOD'95)*, pages 175–186. ACM Press, May 1995.

[PMS97] M. J. Pazzani, S. Mani, and W. R. Shankle. Comprehensible knowledge-discovery in databases. In *Proceedings of the 19th international Conference of the Cognitive Science society (COGSCI'97)*, pages 596–601. Lawrence Erlbaum, August 1997.

[PS91] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI Press, 1991.

[PSF91] G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI Press, 1991.

[PSM94] G. Piatetsky-Shapiro and C. J. Matheus. The interestingness of deviations. In *AAAI'94 Workshop on Knowledge Discovery in Databases*, pages 25–36. AAAI Press, July 1994.

[Qui93] J. R. Quinlan. *C4.5 : Programs for machine learning*. Morgan Kaufmann, 1993.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proceedings of the 21st international conference on Very Large Data Bases (VLDB'95)*, pages 407–419. Morgan Kaufmann, September 1995.

[SA96a] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of Data (SIGMOD'96)*, pages 1–12. ACM Press, June 1996.

[SA96b] R. Srikant and R. Agrawal. Mining sequential patterns : Generalisations and performance improvements. In *Proceedings of the 5th biennial international conference on Extending Database Technology (EDBT'96)*, Lecture Notes in Computer Science, Vol. 1057, pages 3–17. Springer-Verlag, March 1996.

[SAD+93] M. Stonebraker, R. Agrawal, U. Dayal, E. J. Neuhold, and A. Reuter. DBMS research at a crossroad : The Vienna update. In *Proceedings of the 19th international conference on Very Large Data Bases (VLDB'93)*, pages 688–692. Morgan Kaufmann, August 1993.

[SBM98] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets : Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1) :39–68, January 1998.

[SBMU98] C. Silverstein, S. Brin, R. Motwani, and J. D. Ullman. Scalable techniques for mining causal structures. In *Proceedings of the 24th international conference on Very Large Data Bases (VLDB'98)*, pages 594–605. Morgan Kaufmann, August 1998.

[SG99] F. Sha and G. Gardarin. Gradual clustering algorithms for metric spaces. In *Actes des 15èmes journées Bases de Données Avancées (BDA'99)*, pages 305–321, Octobre 1999.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in larges databases. In *Proceedings of the 21st international conference on Very Large Data Bases (VLDB'95)*, pages 432–444. Morgan Kaufmann, September 1995.

[Sri96] R. Srikant. *Fast algorithms for mining association rules and sequential patterns.* PhD thesis, University of Wisconsin, 1996.

[SSU91] A. Silberschatz, M. Stonebraker, and J. D. Ullman. Database systems : Achievements and opportunities. *Communications of the ACM*, 34(10) :110–120, October 1991.

[ST95] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the 1st international conference on Knowledge Discovery and Data mining (KDD'95)*, pages 275–281. AAAI Press, August 1995.

[ST96] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6) :970–974, December 1996.

[Stu97] G. Stumme. Concept exploration - a tool for creating and exploring conceptual hierarchies. In D. Lukose, H. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *Conceptual structures : Fulfilling Peirce's dream*, Lecture Notes in Artificial Intelligence, Vol. 1257, pages 318–331. Springer-Verlag, 1997.

[SVA97] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings of the 3rd international conference on Knowledge Discovery and Data mining (KDD'97)*, pages 67–73. AAAI Press, August 1997.

[SW85] A. Shoshani and H. K. T. Wong. Statistical and scientific database issues. *IEEE Transactions on Software Engineering*, 11(10) :1040–1047, October 1985.

[SW95] G. Stumme and R. Wille. A geometrical heuristic for drawing concept lattices. In R. Tamassia and I.G. Tollis, editors, *Graph Drawing*, Lecture Notes in Computer Science, Vol. 894, pages 452–459. Springer-Verlag, 1995.

[Tao00] R. Taouil. *Algorithmique du treillis des fermés : application à l'analyse formelle de concepts et aux bases de données*. PhD thesis, Université de Clermont-Ferrand II, 2000. En préparation.

[TKR+95] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping discovered association rules. In *ECML'95 MLnet workshop on statistics, machine learning, and knowledge discovery in databases*, pages 47–52, April 1995.

[Toi96a] H. Toivonen. *Discovery of frequent patterns in large data collection*. PhD thesis, University of Helsinki, 1996.

[Toi96b] H. Toivonen. Sampling large databases for association rules. In *Proceedings of the 22nd international conference on Very Large Data Bases (VLDB'96)*, pages 134–145. Morgan Kaufmann, September 1996.

[TPBL99] R. Taouil, N. Pasquier, Y. Bastide, and L. Lakhal. Computing closed sets lattices : Algorithms, applications and performances. Rapport de recherche. Soumis pour publication, 1999.

[TPBL00] R. Taouil, N. Pasquier, Y. Bastide, and L. Lakhal. Mining bases for association rules using Galois closed sets. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'2000)*, page 307. IEEE Computer Society Press, Febuary 2000.

[Uth96] R. Uthurusamy. From data mining to knowledge discovery : Current challenges and future directions. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 561–569. AAAI Press, 1996.

[Wai98] K. Waiyamai. Object modeling and querying of concept lattice based applications. In *Actes du 16ème congrès sur l'Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'98)*, pages 437–455, May 1998.

[Wai99] K. Waiyamai. *Découverte et gestion de connaissances dans les bases de données (Data Mining) : une approche par treillis de concepts fréquents.* PhD thesis, Université de Clermont-Ferrand II, July 1999.

[Wid95] J. Widom. Research problems in data warehousing. In *Proceedings of the 4th international conference on Information and Knowledge Management (CIKM'95)*, pages 25–30. ACM Press, November 1995.

[Wil82] R. Wille. Restructuring lattices theory : An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, pages 445–470. Reidel, Dordrecht-Boston, 1982.

[Wil92] R. Wille. Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications*, 23 :493–515, 1992.

[WMS+94] J. T.-L. Wang, G. W. Marr, B. Shapiro, D. Shasha, and K. Zhang. Combinatorial pattern discovery for scientific data : Some preliminary results. In *Proceedings of the 1994 ACM SIGMOD international confe-*

rence on Management of Data (SIGMOD'94), pages 115–125. ACM Press, May 1994.

[WTL97] K. Waiyamai, R. Taouil, and L. Lakhal. Towards an object database approach for managing concept lattice based applications. In *Proceedings of the 16th international conference on Conceptual Modeling (ER'97)*, Lecture Notes in Computer Science, Vol. 1331, pages 299–312. Springer-Verlag, November 1997.

[WTL98] K. Waiyamai, R. Taouil, and L. Lakhal. Querying of concept-lattices in object databases. In *Proceedings of the 3rd biennial international conference on Integrated Design and Process Technologiy (IDPT'98)*, pages 280–289, July 1998.

[Zak98] M. J. Zaki. *Scalable data mining for rules.* PhD thesis, University of Rochester, 1998.

[ZHL+98] O. R. Zaïane, J. Han, Z.-N. Li, S. H. Chee, and J. Y. Chiang. MultimediaMiner : A system prototype for multimedia data mining. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of Data (SIGMOD'98)*, pages 581–583. ACM Press, June 1998.

[ZO98] M. J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *DMKD'98 workshop on research issues in Data Mining and Knowledge Discovery*, pages 1–8. ACM Press, June 1998.

[ZPOL97] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proceedings of the 3rd international conference on Knowledge Discovery and Data mining (KDD'97)*, pages 283–286. AAAI Press, August 1997.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. Birch : An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of Data (SIGMOD'96)*, pages 103–114. ACM Press, June 1996.